PI SENSORS
Thermal monitors for hot projects

UBUNTU FILES
Master file ownership and network shares

SEEDBOXES
Stream and share at super speeds online

LINUX FORMAT

The #1 open source mag

HOME STREAMING
TESTED: The top 5 apps to enjoy your media at home

FORTRESS LINUX!

Protect your privacy with Qubes

- Virtually isolated apps
- Multiple OS support
- Tor secured network
- Self-destruct mode

100 pages of Linux tricks, tips & more!

PLUS: HOW TO

- Master the oldest chat system: IRC
- Get the best 4K display upgrade
- Add history to the LXF Shell

AURORA RADIO
How to tune in to the music of the sun's rays

EASY OS CHECK
Code an in-depth OS health check tool

GOOGLE MATTER
Discover and build the new internet of things

LXF November 2024
FUTURE

LINUX FORMAT

Corporate Patron Free Software Foundation

MEET THE TEAM

This issue, we're setting up Qubes OS for total privacy. What little thing do you do to help bolster your privacy or security either on or offline?

Jonni Bidwell
Running a *Pi-hole* gateway (either on a Pi or in the cloud) is not as scary as it sounds. It sinkholes DNS requests for known ad networks, enabling you to browse ad-free on mobile devices and smart TVs. A modicum of privacy is then clawed back from the pesky ad networks.

Michael Reed
Years ago, I was experimenting and used password-based SSH that was temporarily accessible from the outside world. Then *[look of shame]* I forgot about it. A few days later, I checked the SSH log and there were hundreds of login attempts! Lesson learned.

Mayank Sharma
Privacy-respecting browsers, secure VPNs, nuking cookies and cache, complex passwords, and a host of other exercises make up my privacy regimen. Of course, I just as frequently resort to the less popular alternative – not giving a damn, but coupled with fervent prayer to keep evil at bay.

Les Pounder
We live in the public spotlight, sharing photos and stories via social media. I try to limit how much I share online, going as far as making sure that pictures only show the subject, and not the surroundings. Which really killed my photography business in the Lake District!

Nick Peers
I've long preferred self-hosted services to the cloud for privacy and security reasons. Now I've deployed a Wireguard VPN server, which allows me to access certain services like my Bitwarden password vault remotely without exposing them to the internet. Win-win.

*Savings are based on the cover price.

Reassuringly expensive

Linux Format has cost £6.49 for over 18 years but finally, after the world going online, a pandemic, oil price spikes, a cost of living crisis and various recent paper production strikes (we're looking at you, Finland), we're having to put the price of the magazine up to £6.99. There's no sugar-coating a price increase but it guarantees we can be here for a while longer – and, as always, if you subscribe, you can enjoy 50% off that cover price. While also dodging having to go out in the wind and rain, as we'll post it direct to you!

Hopefully you'll keep on feeling *Linux Format* is worth the price tag, bringing you hot topics such as this issue's cover feature on the latest release of Qubes – the reasonably secure OS. The Fort Knox of the OS world, it fully isolates all running programs, colour-codes them depending on trust, and encrypts network connections over Tor. If you're the paranoid type, it can really help soothe your privacy worries!

After the excitement of the Pi Pico 2 release, we have a double dose of Les Pounder looking into Pico sensors and LEDs. Seedboxes are on our minds for faster sharing, plus if you like to share media around your home, we round up the best server software for streaming and sharing more things to enjoy!

Neil

Neil Mohr Editor
neil.mohr@futurenet.com

Contents

SUBSCRIBE NOW!
Page 16

REVIEWS

BUILD YOUR LINUX FORTRESS!

Chronically insecure and unreasonable Jonni Bidwell needs reasonable operational security. With Qubes, and containment and isolation, he finds a solution.
See page 32!

CREDIT: Magictrorch

ROUNDUP

IN DEPTH

CREDIT: Fotolia/saniphoto 2008

CONTENTS

Pi USER

CODING ACADEMY

REGULARS AT A GLANCE

TUTORIALS

CREDIT: https://www.pexels.com/Stein Egil Liland

ADMINISTERIA

CREDIT: Getty Images/ Bill Hinton Photography, Giga-Rapid/Nate

Newsdesk

THIS ISSUE: News piped from Plumbers Conference » Religion of Rust » Snapdragon support stepped up » Linux goes real-time

PLUMBERS CONFERENCE

Linux summit points to Rust and RISC future

Annual Linux Plumbers Conference featured hours of discussion on topics such as Rust and RISC-V adoption, as well as kernel maintenance.

The 2024 Linux Plumbers Conference (LPC) ran from 18th to 20th September. The annual event, arranged by the Linux Foundation, brings together all developers working at the 'plumbing layer' of the OS, along with other interested parties.

The event in Vienna's Austria Center hosted a number of microconferences on a diverse range of topics, including System Boot and Security, Zoned Storage Devices and RISC-V.

Linus Torvalds was in attendance as part of the open source summit. The keynote discussion took place with Verizon's head of open source, Dirk Hohndel.

Topics of discussion included the latest release of Linux Kernel 6.11. Torvalds deprecatingly stated that releases "are not supposed to be exciting", claiming that the fun comes the day after a new version is made available, as developers can begin preparing code for the next.

Hohndel also raised the changing nature of the kernel, given that he was one of the first developers to work on it.

Torvalds admitted, "These days, more than half the kernel is driver support." He cited one of the strengths of kernel development is that different people care about "very specific architectures and filesystems". He used this perspective to toe the line on the Rust versus C++ debate, claiming he doesn't see integration of the newer programming language as a failure.

Torvalds went on to praise the many people who have assisted him in kernel development, which prompted Hohndel to ask about the Maintainer Summit, which also took place at LPC.

The recurring topic that comes up every year is that of developer burnout. After joking that grey seemed to be the prevailing hair colour for maintainers this year, Torvalds candidly admitted that this is an issue.

However, he also went on to praise maintainers who have been successfully contributing to Linux for over 30 years, citing the huge number of open source projects that are still being released every few months.

Hohndel also raised the spectre of recent layoffs at big tech companies, some of which have "aggressively" funded open source. Torvalds pointed out that when he began development, open source was virtually unheard of. He also highlighted what he saw as the "cyclical" nature of companies hiring and firing staff.

He concluded by emphasising how open source has become "ubiquitous" to the extent that new programmers can use it as a way to enter the industry.

Full details of all conference events are available from **https://lpc.events/**. The LPC YouTube Channel (**https://www.youtube.com/@LinuxPlumbersConference**) also has recordings of all the conferences in each room for the 2024 events.

Torvalds said he preferred the Q&A format with Hohndel over a keynote address, as he feels nervous about public speaking.

The Linux Plumbers Conference took place across three days, with multiple events in rooms at the Austria Center in Vienna.

CREDIT: YouTube/The Linux Foundation, LPC

PROGRAMMING WARS!

Rust vs C out of control

The promise of memory-safe Rust code provokes ugly debates leading to police intervention.

During the keynote discussion at LPC 2024, Dirk Hohndel asked Linus Torvalds about the recent decision by Microsoft engineer Wedson Almeida Filho to step down as a Rust maintainer.

While assuring the Rust Linux team they were great, Filho explained in a post on the Linux Kernel mailing list: "After almost four years, I find myself lacking the energy and enthusiasm I once had to respond to some of the nontechnical nonsense."

Filho cited a presentation where Ted Tso accused him of trying to convert everyone to the "religion of Rust".

His post contains a link to a YouTube video of a presentation he did with Kent Overstreet on Rust for filesystems. It shows the attendees engaging in an aggressive line of questioning, going so far as to claim there are efforts to "switch everyone over to the religion" of Rust.

Filho ended his post with the same message he emphasised in the presentation: that he wasn't trying to force anyone to learn the Rust language nor prevent refactorings of C code.

Rene Rebé, a well-respected Linux maintainer and contributor, also seemingly fell victim to what Filho described as nontechnical nonsense in mid-September. While live streaming a development session for his own OS T2 Linux, he fell victim to a 'swatting' attack.

Police knocked on his door, and he was placed in cuffs and questioned for an hour before being released. There's no clear proof that the attack was initiated by a member of the Rust development community but Rebé has criticised the project in the past.

At the summit, Torvalds said he wasn't sure why Rust was so contentious, as the debate had "taken on certain religious overtones".

HARDWARE

Snapdragon support improving

ARM64 image of Ubuntu 24.10 to support X13s.

To date, Ubuntu has tentatively supported the ThinkPad X13s ARM laptop but it's been tricky, tinkering with internal settings and external software.

In welcome news to Lenovo lovers, Ubuntu 24.10 can now be installed automatically on the notebook, which uses the Snapdragon 8cx Gen 3 SoC. This update simplifies installation using the generic ARM64 image, which can be booted and installed via USB in exactly the same way as for x86_64 images.

In order for the installation to function correctly, users have to choose Install Third-Party Software For Graphics And Wi-Fi during setup. This enables and downloads the relevant packages for the X13s, which includes the firmware binaries for hardware acceleration and webcam support via the latest Linux kernel (6.11).

This development is the result of months of hard work from the Ubuntu community, but compatibility issues remain with the Ubuntu 24.10 ARM64 image, including variable audio performance, buggy virtualisation/hibernation and a non-functioning fingerprint reader.

While there's still work to be done, and this particular model is no longer sold by Lenovo, the developments bode well for ARM64 support for the Linux ecosystem, setting the stage for more ARM-based laptops running Linux. The latest version of the kernel also partially supports the Snapdragon X Elite series. X13s owners wanting to try Ubuntu 24.10 can download images from **http://cdimage.ubuntu.com/daily-live/current/**.

OPINION

MANY TONGUES

Italo Vignoli is one of the founders of LibreOffice and the Document Foundation.

"*LibreOffice* is the desktop software available in the largest number of language versions, thanks to an army of nearly 1,000 volunteer translators who've localised the user interface strings into 163 languages over the past five years, totalling more than 30 million changes.

Of these 163, 120 have been officially released by the Document Foundation. These include regional languages such as Breton and Venetian, and smaller languages spoken by only part of the population and in danger of extinction, such as Guarani in Paraguay.

Offering *LibreOffice* in a large number of languages to meet the needs not only of users in countries served by proprietary software companies, because it offers a significant return on investment, but also of users in less affluent countries, neglected by proprietary software companies, is a great strength of our project.

A strength that is probably little known or understood by those who can count on the localisation of all programs because they live in a so-called rich country. A strength that makes us very proud of our community, which is capable of doing something extraordinary thanks to the people who give up their time to translate *LibreOffice*."

OPINION

REDISCENT EVILS!

Dave Stokes
is a technology evangelist at Percona.

Six months ago, you would never have heard of Valkey. It is a widely used and heavily featured high-performance key/value data store. In March 2024, Redis decided to forsake its open source licence for something more restrictive that would – it hoped – turn the revenue spigot to full.

The name Redis originated from the Remote Dictionary Server. It found favour with GitHub, Instagram and more. Thousands contributed to the codebase. Many used it to cache data and as a fast in-memory data store. However, the change to a Server-Side Public License made many unhappy. This reverse Robin Hood move was beyond unpopular.

The Redis code was forked and named Valkey. The Linux Foundation adopted it. Amazon Web Services, Percona, Google Cloud and Ericsson support it. There is even a Valkey Developer Day at the Open Source Summit Europe in Vienna.

This is a shining example of how open source is supposed to work. Valkey demonstrates how to launch a new project cooperatively, add impressive features, and deliver better performance. Doing that in six months is fantastic.

You may not have heard of Valkey six months ago, but now you have. Will anyone remember Redis in six months?

KERNEL

Linux is finally a real-time OS!

The culmination of decades of hard work was realised in September.

Thomas **G**leixner bowed graciously as he presented the written pull request to enable **PREEMPT_RT** to a highly amused **L**inus **T**orvalds.

After over 20 years and thousands of lines of code, the Linux kernel has achieved hard real-time capabilities, a holy grail that some developers never believed would be achieved in a general-purpose OS.

On 19th September, Thomas Gleixner humorously honoured the occasion by handing Linus Torvalds the pull request wrapped in gold and tied with a ribbon. This marked the completion of the real-time pre-emption patches, allowing future kernel versions to handle tasks with guaranteed response times, making Linux suitable even for demanding environments such as self-driving vehicles or high-performance media processing.

Torvalds acted on the pull request the following morning. Still, these new capabilities are scheduled to be available in kernel v6.12 on November 25th, so aren't yet a part of current Linux distros. But thanks to this new real-time support, Linux should be able to guarantee response times of 100 microseconds, far shorter than the years of development that went into achieving this milestone.

CREDIT: kernel.org

GRAPHICS

Microsoft adopts SPIR-V

In a surprise move, Microsoft moves to the open format.

Microsoft has announced plans for DirectX to adopt SPIR-V as its shader interchange format. Support will be integrated into DirectX 12 with the upcoming release of Shader Model 7. The previous format DXIL (DirectX Intermediate Language) is seemingly to be retired.

In its announcement, Microsoft emphasised its commitment to open development processes, stating, "It is our mission that HLSL [High-Level Shading Language] be the best language for compiling graphics and compute shaders for any device or GPU runtime API."

Microsoft is offering translation tools between SPIR-V and DXIL that will facilitate the process.

SPIR-V could become the de facto standard for **API** developers.

SOFTWARE

MS hands over Mono to Wine

Microsoft donates Mono to Wine, urging migration to modern .NET.

Microsoft has announced the decision to transfer stewardship of the Mono project to the Wine Community. The open source framework, which was pivotal in bringing .NET to non-Windows systems such as Linux, last had a major release in 2019.

WineHQ will now manage the Mono project's upstream code. The project repository still exists and Microsoft has promised to make binaries available for up to four years.

Mono was already under the MIT License and Wine already has its own Mono engine (wine-mono), so no drastic immediate changes in the compatibility layer are expected.

Microsoft engineer Jeff **S**chwartz posted on the **M**ono **G**itHub page, thanking developers and recognising the project as a trailblazer for **.NET** implementation.

Distro watch

What's behind the free software sofa?

REDOX OS 0.9.0

Redox is a Unix-like general-purpose microkernel-based OS written in Rust. It incorporates an optional GUI, Orbital, and is source compatible with Linux/BSD programs. This first release includes COSMIC desktop apps such as *Files, Editor* and *Terminal,* and a number of stability and performance improvements, particularly for context switching and on-demand paging. There's also better compatibility with ARM64 devices, but USB support is a work in progress. Read more: **www.redox-os.org**.

This is the first **R**edox release, an **OS** written in **R**ust.

HAIKU R1-BETA5

Anyone who remembers the good old days of BeOS will find this OS hauntingly familiar. The 32-bit images of Haiku, now available in beta, in fact retain binary compatibility with BeOS 5. This release incorporates a number of improvements over the alpha, including simplified colour selection and a dark mode. The desktop bar now automatically displays power settings for portable devices, and USB audio devices are now supported. The developers are also including TUN/TAP to support VPN connections. You can learn more at **www.haiku-os.org**.

Haiku has numerous networking improvements.

MX LINUX 23.4

The MX OS is the result of a collaborative effort between the former antiX and MEPIS development communities. The OS is based on the latest stable version of Debian (in this case v12.7). The default desktop is Xfce and in this latest version, code-named Libretto, the core packages have been updated to 4.18. The package installer has been overhauled and there's better support for mounting and encrypting USB drives. Discover more at **https://mxlinux.org**.

The latest release includes many bug and compatibility fixes.

SDESK ISO 19.1

If you read our in-depth review in **LXF313**, you already know that SDesk is based on Arch Linux and uses Gnome with Wayland sessions as a desktop environment. The latest point release includes a number of new drivers for touchscreens and fingerprint readers. The default *Swirl* browser has an upgraded user interface. While setup is still mostly handled by *Calamares,* certain aspects, such as user account creation, are now managed by the *Gnome Initial Setup* application. You can learn more at **https://stevestudios.net/sdesk**.

SDesk includes an overhauled setup as well as new drivers.

FREEBSD 13.4

FreeBSD follows a rather unusual release cycle in that different versions are made available at different times. This one represents the latest maintenance release for the 13.x series, so the focus is more on bug fixes and driver updates over new features. For instance, a number of packages have been upgraded to the latest versions, including **openssl**, **sqlite3** and the **clang** compiler. There are also stability fixes for native and LinuxKPI-based wireless drivers. See **www.freebsd.org**.

This latest release includes bug fixes and driver upgrades.

OPINION

SMART AUDIO

Julian Bouzas
is a senior software engineer at Collabora.

"The recent release of WirePlumber 0.5, PipeWire's session manager, brought a neat new feature called the Smart Filter Policy.

In PipeWire, audio filters are represented as pairs of two nodes: a virtual client and a virtual device node. Users can control which is the real device linked to the filter's output and which application sends audio to the filter's input using existing GUI tools, such as *PulseAudio Volume Control.*

Up until now, audio filter nodes were always treated like regular nodes in WirePlumber. This meant that if you wanted to use a filter with a specific device, you would have to manually ensure the filter's output was linked to that device and that application streams were linked to the filter's input.

Smart filters automatically and transparently insert themselves in between client streams and devices, so you can have filter configurations tied to specific devices without explicitly redirecting applications or manually moving filters when devices are hotplugged.

This has a profound benefit, as systems can be pre-configured to apply the correct filter only when devices for which it makes sense are selected. The app does not have to know what the input or output is and needn't request specific filters."

OPINION

RT ARTISTS

Jon Masters is a kernel hacker who's been involved with Linux for over 22 years, and works on energy-efficient Arm servers.

"The Real Time (RT) patches have finally been merged into upstream Linux, but there's so much more to this story than just a bunch of code getting merged into the kernel after a long time.

For starters, there are many people to thank for the work, some of whom are no longer with us. Among those is Doug Niehaus, formerly a professor from the University of Kansas whose pioneering work on KURT (Kansas University Real-Time) was a big influence on the work that would follow.

Then there are those who were influential early on, such as Victor Yodaiken, who created an RTLinux project in the '90s. This splits the system into a small Real Time OS that runs real-time tasks, and the Linux kernel, which it runs as a pre-emptible process. This means that software running on RTLinux must be modified to be aware of the environment and its constraints. By contrast, the modern Real Time code is just an extension of Linux, and can run unmodified application software.

Finally, there's Ingo Molnar, Thomas Gleixner, and the early contributions of others (such as Sven Thorsten-Dietrich) that started the modern development effort."

Kernel Watch

Jon Masters summarises the latest happenings in the Linux kernel, so that you don't have to.

Linus Torvalds released the first Release Candidate (RC) kernel for what will become Linux 6.12 in the next couple of months. Among the new features in this kernel is a faster implementation of the getrandom system call using the kernel vDSO (virtual Dynamic Shared Object) mechanism that will speed up a surprising number of applications that require frequent random numbers (and now can skip a true system call). 6.12 also adds support for protected KVM (pKVM) guests, a mechanism used on Android devices to support virtualisation while handling some of the unique constraints of such devices.

About damn time (for Real Time)

Probably the most significant new feature in 6.12 is official support for PREEMPT_RT, aka Real Time, after more than 20 years (yes, really!) of active development. More specifically, Linus finally merged the last blocker (fixes to the kernel's printk logging infrastructure) required to enable PREEMPT_RT in the kernel configuration files on the architectures that support it (Arm, RISC-V and x86). The RT patches enable Linux to provide deterministic real-time guarantees about certain worst case latencies throughout the system, such that (for example) an industrial laser-wielding robot might be expected to respond to changes in its environment before doing damage.

Typical Linux (and Unix, Windows or Mac OS) systems are not real time. Instead, they operate on a best effort basis. If there are 100 tasks that need to run, the kernel does its best to be fair, giving each a slice of time on the available CPU resources to make progress. These may (frequently) be interrupted by the need to process hardware interrupts (messages from hardware devices such as network adaptor cards with packets to process), and such delays can be extremely lengthy (introducing milliseconds of jitter, noticeable even in VoIP calls or video game frame rates sometimes).

Real Time kernels are different. They treat almost anything as a (pre-emptible) process (hence "preempt-rt" in the RT patch names) that can be interrupted by something more important, even the processing of hardware interrupts. The mechanics are very involved (and quite academic), as they need to handle concepts such as priority inversion, in which a lower priority task acquires a lock that a higher priority task is waiting for. Hence the RT patches have fun tricks like priority boosting to temporarily give such an inversion a fix by giving more time to the lower priority task.

Getting the patches into upstream has been a 20-plus-year odyssey. In that time, a number of commercial Linux distros have shipped RT versions. Now that RT is finally fully upstream, expect to see it adopted more broadly. For most day-to-day use, you shouldn't notice much difference, but those who need determinism (audio files and gamers) will benefit. LXF

» ONGOING DEVELOPMENT

The Linux kernel community may feature quite a few professionally employed developers working for major corporations, but it also still relies (heavily) upon volunteer contributions. These include both documentation and regression tracking. Those leading both such efforts have let it be known this month that they may not have the bandwidth and resources to continue without outside help.

A number of comments on the kernel mailing list about the AMD GPU drivers have mentioned stability problems, while Hans de Goede has a blog post up noting that the drivers are large enough that they may take longer than 10 seconds to load on older machines, causing the "plymouth" boot graphics to timeout: **https://hansdegoede.dreamwidth.org/28552.html**.

Work continues to enable Rust language support (for example, in interrupt handlers and loadable modules), in spite of a little drama as several maintainers have stepped down. Linus recently spoke (again) in defence of giving Rust a chance, noting that a lot of folks may be set in their ways with C but that there are good reasons to investigate memory safety.

Answers

Got a burning question about open source or the kernel? Whatever your level, email it to **answers@linuxformat.com**

Neil Bothwick thinks he was an Enigma machine in a past life.

Q Fast font fiddling

Is there any quick single-click way to switch between two fonts (with different sizes) when writing *LibreOffice* documents? I normally use Liberation Serif (12pt) but often switch into Liberation Mono (11pt) to clarify things like program output or user commands used in the middle of other text. Some kind of single-click method would make the writing more fluent.
Yrjö Seppä

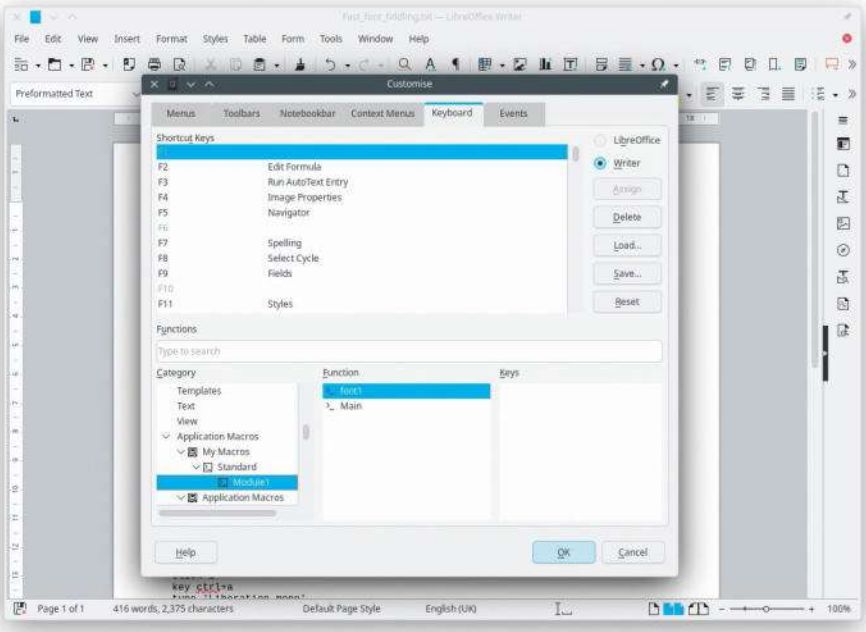
A There are two ways to approach this. One is to use the macro recording facility to record the font change, then bind that to a key. First enable macro recording under Tools > Options > Advanced. Then select Tools > Macros > Record Macro, perform the actions you want, click Stop Recording and save the macro. Now you can assign this to a hotkey by selecting Tools > Customise and going to the Keyboard tab. Go to Application Macros in the Category list and choose your macro. Click on an unused key in the Keys list and press Assign. Now you should have a hotkey to do just what you want. Recording and running macros in *LibreOffice* relies on a compatible version of Java being available. If this does not work for you, try the following alternative.

The other way is to use *xdotool* (or an equivalent). *Xdotool* sends mouse and keyboard events directly to the system. This means it will only work reliably for your needs if the selection boxes for the font are always at the same screen coordinates, which would be the case if you run *LibreOffice* in a full-size window. Then you can write a short *xdotool* script and save this in a file – say, **mono.xdo**:

```
mousemove --sync 480 150
click 1
key ctrl+a
type 'Liberation mono'
key Enter
mousemove --sync 750 150
click 1
key ctrl+a
type '11'
key Enter
```

You have to determine the exact positions to click on by trial and error, depending on your screen and font settings. Bind **xdotool /path/to/mono.xdo** to a hotkey and you are good to go. Make a copy of the script and change the font name and size to select your serif font.

It should be possible to manage this with a window that is not full size by using *xdotool* to find the window, get the position and then calculate the required click locations. The man page contains comprehensive details of all the options if you want to explore that route.



Once you have recorded a macro in LibreOffice, you can attach it to a hotkey for rapid access.

Q Airgapped updates

How do you install programs on PCs with Ubuntu 22.04 or 24.04 that do not have internet connections? It is simple to install Ubuntu, AppImages and Flatpaks but I refer to non-AppImage/Flatpak file types such as DEB and TAR.GZ. An example of a program I downloaded and want to install: *dconf-editor-master.tar.gz (v45.0)* from http://gitlab.gnome.org/GNOME/dconf-editor.
Chris

A This can be tricky. Operating systems tend to assume internet connectivity these days. When you download a source code tarball, like that you mention, you should unpack the archive and look for files called **README** and **INSTALL** (or similar). These detail the steps needed to compile and install the source code. To install from source, you need a compiler toolchain installed. With Ubuntu and its derivatives, you should install the **build-essential** package.

Installing from DEB packages is a different matter. You rarely install a single package; it usually comes with a list of dependencies, all of which need to be downloaded. You can use the *APT* command-line package manager to create a list of packages to download, like this:

```
$ sudo apt-get install somepkg --print-uris >uris.txt
```

This creates a list of download locations and where they should be downloaded to (sometimes *APT* wants to save a file under a different name to the source). You can use *AWK* to turn this into a script to download what you need:

```
$ awk '/http/ {print "wget -q", $1, "-O", $2}' uris.txt >download.sh
```

This a capital letter O, not a zero. Copy this to a removable drive and run the script on an internet-connected computer:

```
$ sh download.sh
```

Now you can copy the downloaded files to **/var/cache/apt/archives** on the original system and run `apt-get install` again without the **--print-uris** option.

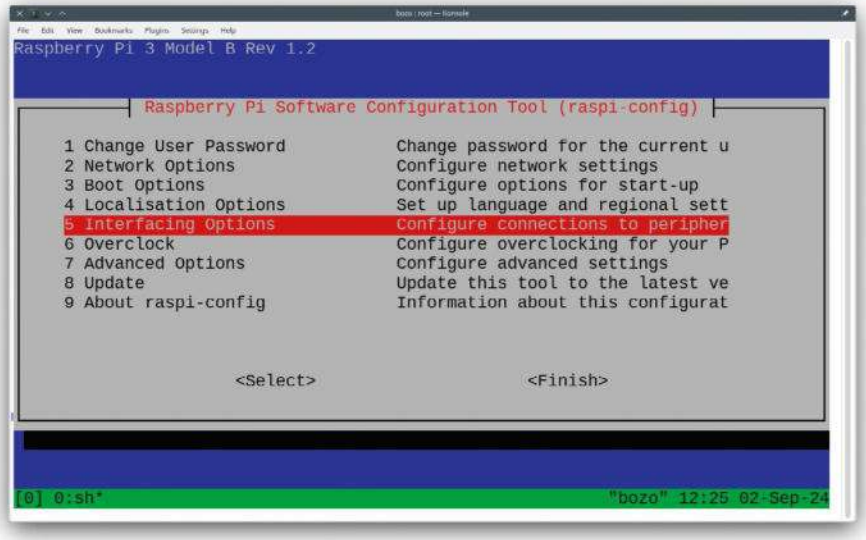
This works to an extent, but if you do not update the lists used by the package manager, some of the DEB files you need to download may not be available as they have been superseded. This is particularly true of packages that have been updated for security reasons.

You can download new index files in a similar way to how you get the packages. This command generates a script that will download the package indices:

```
$ apt-get indextargets --format 'wget $(URI)' >indices.sh
```

Run that from a removable drive, then copy downloaded files to **/var/lib/apt/lists**.

It is a convoluted process. One way of avoiding it is if you are using a laptop. Then you can create a virtual machine that is identical to the system you wish to update. Go somewhere with Wi-Fi and run the updates or installs you need, then copy the package and index files from the VM to the other computer when you get back.



No display on your Raspberry Pi? You can get in with SSH to run raspi-config.

Q Waylaid by Wayland

I upgraded a Pi 4 to Bookworm 5.3, a fresh download to a new microSD card. The Pi 4 failed to boot. Although this monitor, a VILVA model V156F1, was working fine with the Pi 4 running Buster, I suspected something was wrong with it, so connected it to an Acer model AL1914 and it booted with no problem. Then I connected the VILVA monitor to a Pi 400 and it worked fine. My conclusion was that Wayland was the culprit. Is there a config file to edit manually so the Pi can boot into X11 rather than Wayland when *raspi-config* can't be run, as you can't run *raspi-config* when the Pi doesn't boot.
Don Dollberg

A It does seem like Wayland is not correctly detecting the monitor. The documentation on this is rather fragmented, but it appears there are two solutions, and you may need to use both of them. The first is to edit **/boot/firmware/cmdline.txt** and add the line:

```
wayland=off
```

If there is already a line enabling Wayland, edit that instead, changing **on** to **off**. The second step involves running *raspi-config*. You say the Pi is not booting but it may be running, just with no display. If you know the IP address or hostname of the Pi, you can connect via SSH and run *raspi-config*. This is the command to use:

```
$ sudo raspi-config nonint do_wayland W1
```

W1 switches to X11, **W2** goes back to Wayland. This assumes you know the address of the Pi and that the SSH server is running. You can find the address of the Pi with *nmap*. If your network uses the 192.168.1.* address range, use this:

```
$ sudo nmap -sn 192.168.1.0/24
```

Make the obvious change if you use a different address range. You should be able to identify the Pi from the output. If not, run it with the Pi turned off and again with it on and compare the results.

Enabling SSH without *raspi-config* is easy. Mount the boot partition of the SD card on your computer, *cd* to it and run:

```
$ touch ssh
```

This creates an empty file called **SSH** on the card. When the Pi boots, it looks for that file and starts the SSH server. The file is then deleted. The SSH server is enabled for one boot only, but you can enable it permanently in *raspi-config*, which you can run over an SSH session. Now run:

```
$ ssh PI-ADDRESS-OF-PI
```

Log in with your username and password (defaults of **pi** and **raspberry**) to be able to make the changes you need.

» A QUICK REFERENCE TO... COMPILING SOFTWARE

You should use your distro's package manager whenever possible – it keeps your system consistent and up to date – but if you do need to build from source, the process is straightforward.

First unpack the source code archive (or tarball) with:

```
$ tar xvf foo-1.2.3.tar.gz
```

The source is normally unpacked into a directory with a similar name to the tarball. Enter this with `cd foo-1.2.3`. Look for files called **README** or **INSTALL** and read them. They usually contain installation instructions. The standard procedure is:

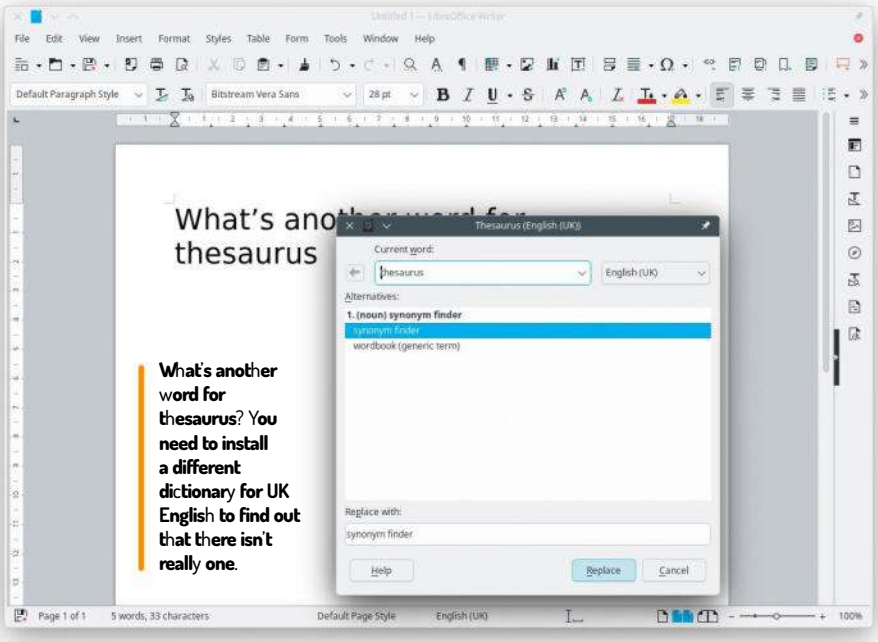
```
$ ./configure
$ make
$ sudo make install
```

The first command checks your system, ensuring you have the necessary dependencies and setting up any option features for the program. It is a good idea to run `./configure --help` first to see the available options.

The second command compiles the software, placing the files it creates in the current directory. It is not necessary to be root to configure or compile, but the third stage copies the compiled files to system directories, so it needs root permissions. This is why we use *su* to run just this one command as root. Ubuntu users should replace this command with:

```
$ sudo make install
```

Unless you told *configure* otherwise, the compiled program is generally installed to **/usr/local/bin**. If you run an RPM system and *configure* complains about a library not being there when you know it is, install the corresponding **-devel** package – for example, libbar-devel.



Q Multiple mounts

I've been trying to add extra hard drives and I'm not having much luck. I can format and mount the drives to folders without any issue. I'm running Zorin and have moved /home to a separate drive from the root drive. I then made the mistake of trying to mount the extra drives to subfolders of /home and the system crashed. So I tried a different approach: I made subfolders in /mnt (/mnt/a, b, etc) and mounted the drives to the subfolders. If I make symlinks from the subfolders to my desktop (`ln -s /mnt/a /home/user/Desktop`) I don't get any permissions to use the folders.
Jay Hawkins

A You have two issues here: mounting the drives and setting permissions. For mounting permanently attached drives (as opposed to removable drives), listing them in **/etc/fstab** is the way to go. You can mount a drive (or more correctly, a filesystem) anywhere you like. You do not say how you tried to mount them under **/home** but this should not cause an issue, and mounting a drive should never crash the system unless you do something rash like mounting it over a critical directory.

The entry for a drive in **/etc/fstab** should look something like:

```
/dev/sdb1 /home/user/data ext4
noatime 0 0
```

The first field is the drive. You can use a dev node, but these are not guaranteed to be the same for every drive every time you boot. It is safer to use a filesystem label or the drive's UUID. See the UUID with:

```
$ lsblk -f /dev/sdb1
```

Filesystem labels, which you can add to a filesystem with `e2label` for ext4 filesystems, make for the most readable **fstab** entries. Then change the first field of the **fstab** entry to one of:

```
UUID=xxxx-xxxx....
LABEL=mylabel
```

The second field is the mount point, under your **home** directory here. Convention is to mount extra fixed drives under **/mnt**, because Linux is a multi-user system, but it is your computer and you can mount them in **home** if you like. The third field is the filesystem type, usually ext4 for Linux . You can use **auto** to have the OS determine the type of filesystem, but it is best to give it explicitly. The fourth field contains mount options for the filesystem, separated by commas: **noatime** is often used to slightly speed up access, put the word **defaults** if you do not want to change any options. Adding **noauto** means the filesystem will not be mounted at boot time. If you do that, also add users so you can mount the filesystem without becoming root.

The lack of permissions is a separate issue. Permissions are stored in the filesystem and have nothing to do with mount point permissions. The root of a new Linux filesystem is owned by the root user, which is why you can't access it as a user. The solution is to change the permissions after mounting the drive:

```
$ chown $USER: /home/user/data
```

You only need to do this once for each filesystem. If you already have content on a drive, add the **-R** option to *chown* to apply the change to all contents, too.

Q Another word please

I am using *LibreOffice* with Linux Mint 21.3 and can find no thesaurus in UK English. From the forums, this seems to be a thorny problem – surely a thesaurus doesn't have to be 100% perfect.
Dave MacFarlane

A *LibreOffice* uses *Hunspell* for spell checking and thesaurus function. *Hunspell* does have an English thesaurus, but it is for US English. It would be nice to be able to use this with UK content – a thesaurus is there to give guidance only, so spelling differences would not matter – but you cannot. If you switch your default language to English (US) in Tools > Options > Language Settings > Languages, the thesaurus is available but you're now using the US spell checker, too. However, there is another dictionary file you can use that includes a UK thesaurus. Download it from **https://bit.ly/LXF321Thesaurus** then go to Tools > Extension Manager in *LibreOffice*, press the add button and select the OXT file you downloaded. Allow *LibreOffice* to restart and you should have a UK English thesaurus available.

There is another option, that isn't tied to use with *LibreOffice*, a thesaurus called *Artha* (**https://artha.sourceforge.net/wiki/index.php/Home**). This is in Ubuntu's repos so should be available for Linux Mint, too. On first run, it sets up a hotkey, usually Ctrl+Alt+W, to call it, but you can change this in the main window. Once running, you only need to select a word in whatever app you are using and press the hotkey. *Artha* opens its window with a list of synonyms.

Thanks for this question. As a result I'm now using *Artha* with my preferred editor and you may see a larger vocabulary/lexicon in *Answers* in future. LXF

GET HELP NOW!

We'd love to try to answer any questions you send to answers@linuxformat.com, no matter what the level. We've all been stuck before, so don't be shy. However, we're only human (although many suspect Neil is Great Old One), so it's important that you include as much information as you can. If something works on one distro but not another, tell us. If you get an error message, please tell us the exact message and precisely what you did to invoke it.

If you have, or suspect, a hardware problem, let us know about the hardware. Consider installing *hardinfo* or *lshw*. These programs list the hardware on your machine, so send us their output. If you're unwilling, or unable, to install these, run the following commands in a root terminal and send us the system.txt file, too:

```
uname -a > system.txt
lspci >> system.txt
lspci -vv >> system.txt
```

Mailserver

WRITE TO US

Do you have a burning Linux-related issue that you want to discuss? Write to us at *Linux Format*, Future Publishing, Quay House, The Ambury, Bath, BA1 1UA or email letters@linuxformat.com.

LoRa, LoRa laughs

As an LXF fan/reader, I'm sending this email to suggest the inclusion of the topic of LoRa, LoRaWAN, LoRaP2P and Meshtastic in some future issue. I strongly believe this is a rich open source topic that LXF readers will probably appreciate. I'm personally interested in its communication capabilities, but also in tracking kids/pets and communicating with them, without forgetting a possible comparative of the many open source but also commercial hardware available out there. I understand that you probably have set the agenda for future LXFs, however I believe it is worth attempting to influence this agenda. Looking forward to hearing from you!

Bernard

Sounds like we should all be communicating with LoRa.

Neil says...

I haven't come across LoRa at all, so thanks for bringing it to my attention. I'm happy to look at any open source hardware solutions, especially if they're easy to integrate and use. I'm sure our man Tam would love to get some kit in his lab to test out.

Started here!

I just got around to start reading LXF317 when I read your welcome column for that issue. I have been a Linux user since 2008. I picked up a Linux magazine that had a version of Ubuntu 8.4 on an accompanying CD and was intrigued about what Linux was all about.

I was getting fed up with Microsoft Windows at the time and decided to try out the CD. I was pleased with what I read and decided to dual boot the Ubuntu OS with Windows XP on a machine I had. It took me about six months to decide to leave behind the Windows OS. It was a little scary when I formatted the entire disk to go to Linux, but I was glad I did.

Through the years, I would read about the latest antics that Microsoft would pull on its customers and would just shake my head. The latest Windows version, number 11, had me really shaking my head as its requirements would force people to give up perfectly good machines that work just fine. Many of those machines will end up in landfills and recycling bins for no good reason.

Thank goodness Linux can come to the rescue and save some desktops from oblivion. I have worked with various versions of Ubuntu, Mint and Puppy Linux on many machines, and for the most part, all versions can run on any machine. I enjoy reading *Linux Format* and I hope you can continue the magazine for the foreseeable future.

Dan Ramos, USA

We hate to say it, but lots of people only use Windows for gaming.

Neil says...

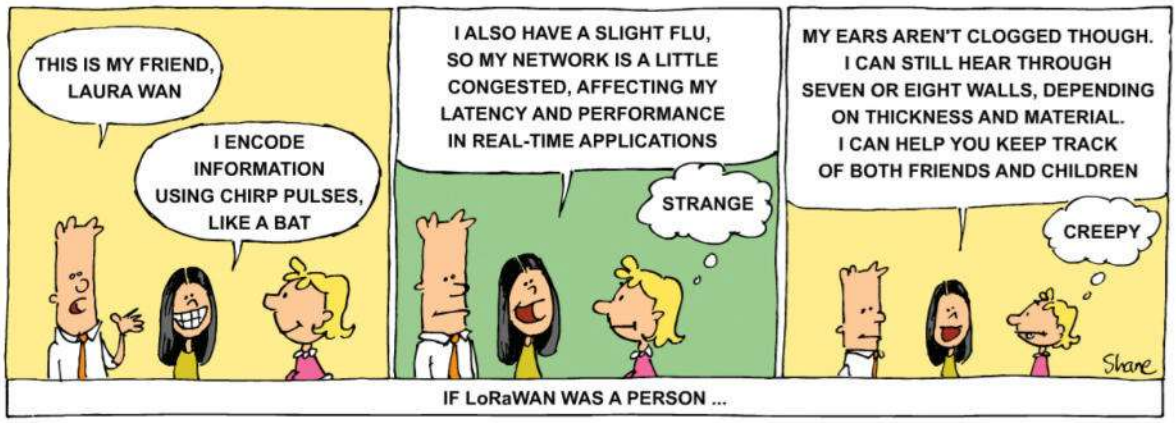
Thanks for the kind words, Dan, and letting us know how you switched to Linux. Microsoft's pushing adverts through Windows 11 does seem to be a real turn off, though I wonder if Google is going to do similar things through Android eventually?

Begin there!

Love your magazine and I'm hoping for some future issues on Rust, using Steam games with Linux (to help make it more popular) and setting up Linux on a VM (I know this but lots of beginners don't, and this is the advice I give them).

Erik

Helpdex



Perhaps we should just stick to using sudo? And, yes, this is the real logo for sudo. Answers on a postcard...

Neil says...
Thanks for the pointers! We're hoping to have a beginner's guide to Rust coming up in a few issues' time, and we've started covering Steam partly through Steam Deck hacks, but it's probably something we should cover now and again, too. I'd expect we'll have an updated feature on VMs down the line, but a beginner's angle might be the thing to do.

LXF is wrong!

I'm concerned about the quality of my preferred Linux magazine as I regularly discover mistakes. The article that made me write is *Linux Basics* in LXF316. For example: "Each flag can be preceded by one (-) or two (--) dashes." Firstly, it's an option and not a flag, and secondly, the name/symbol of the option differs when one (-) or two (--) dashes are used.

How to gain root access: `$ su` is not the correct way to gain root access; `$ su -` is. The `-` is of utmost importance as it will, among other things, also load root's environmental variables such as $PATH, .bashrc and .bash_history of user root.

Changing file permissions: please don't teach beginners to set permissions to 777 and certainly not recursively. Nor set the permissions of a directory recursively to 777.

Keep up the good work and thanks for the interesting articles each month. May I ask you to also think about other distro users – for example, Fedora instead of focusing on Debian-based distros?
Thierry Debaene

Neil says...
You pull up some really good points. I think the first was just laziness, you're right the -- is for the full name of the option; the single is the shorthand. The su thing I'd totally overlooked. That's useful to know for a lazy sudo user. While I'm sure I've previously suggested people never use 777, Nick's article is just an introduction to how these things work, and we should have pointed this out. And yes, more Fedora on the way! LXF

» LETTER OF THE MONTH

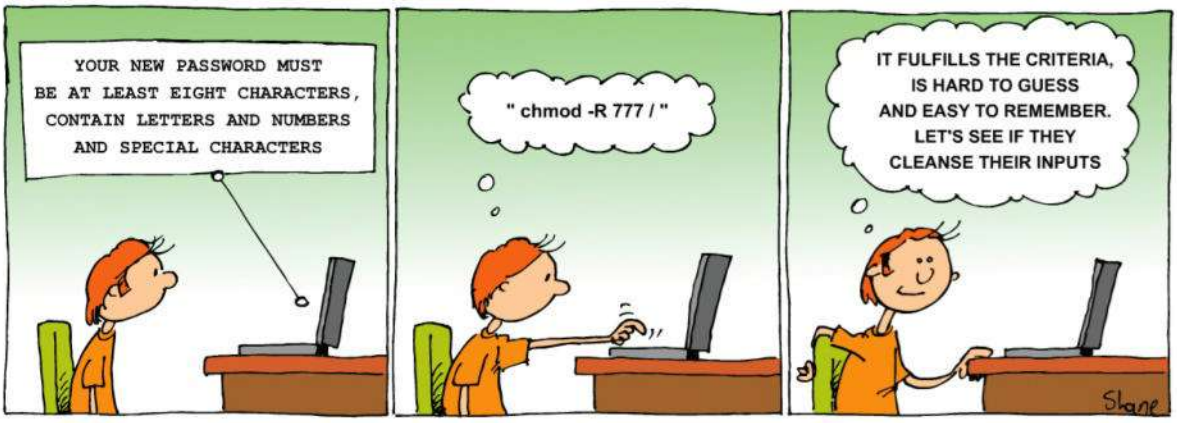
Fedora for ever!

Thanks for the subscribe reminder but I've already resubscribed .I guess your system failed to recognise it as a renewal and treated it as a new subscription. I wish you were still including DVDs with the magazine. And I wish you offered a DVD archive of past issues, as *Linux Magazine* does. I also wish that you covered Fedora better – the best and most advanced Linux distro, which has consistently provided excellent design engineering. IBM is one of the most successful computer companies in the world. It is at the forefront of quantum computing (which will exceed AI in importance). IBM paid $26 billion to purchase Red Hat Linux. Ubuntu is better than Windows, but in my opinion Fedora is better than Ubuntu (with a history of design reversals and poor choices). How long will it be before Ubuntu is only supporting Flatpaks? When Fedora is offering an OS for smartphones, I am confident it won't disappear. When Fedora does it, it usually sticks. As a retired award-winning R&D engineer, it astounds me how wise Fedora design choices have proven to be. I don't have time to learn design choices that won't stick.
Ed Scott

Find out why everyone thinks Fedora is the bee's knees!

Neil says...
Have you considered taking out two subscriptions!? As for a DVD archive, because you're a subscriber, you have full access to the online archive, which is sort of a better thing, we think.

As for Fedora, we're actually covering Fedora 41 in the next issue as our main feature. It's something I've been wanting to do for a while but there's always so many things we want to do! I'm not sure an Ubuntu/Fedora comparison is entirely fair or useful, as Fedora is more an experimental short-term support release, which is not to knock it. Canonical created Snaps as it can support command-line tools for its commercial needs, something Flatpaks don't do, and I think that's why Ubuntu couldn't switch, for better or for worse.



shane_collinge@yahoo.com

YOUR DIGITAL ISSUE ACCESS

Linux Format print subscribers can now access digital back issues two ways!* Who's a lucky bunch of readers?!

» As a big thank you for subscribing to *Linux Format Print Edition*, we've always offered digital access to past issues. It seems The Management thought it was a good idea and has rolled out a new system using Pocketmags!

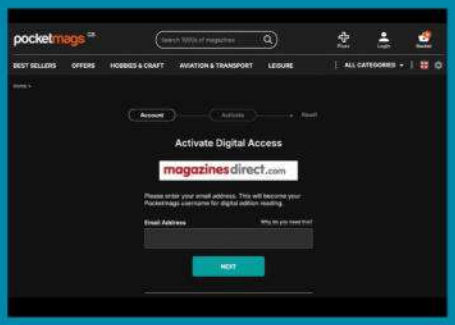
This means there are now two ways of accessing **LXF** back issues for subscribers. The new Pocketmags service goes back to March 2014, including over 140 full issues, while our older PDF archive goes back to **LXF66** (a crazy 8-bit filling 254 issues), though issues before **LXF120** have had covers and ads stripped. We've outlined how to access both below. Enjoy!

*Terms and conditions apply – see page 17 for full details.

FANCY NEW POCKETMAGS

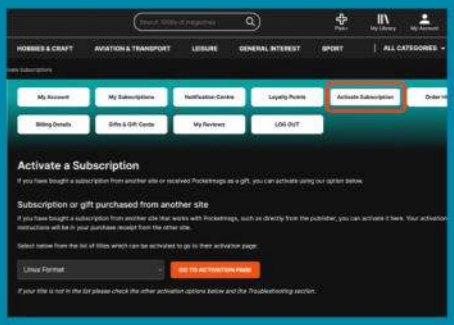
1 Activate your access

Head to **www.magazinesdirect.com/digital-instructions/** and click the Start Reading link that'll transfer you to the Pocketmags activation page.



2 Create an account

Enter the email that was used to create the subscription. You'll be prompted to create a Pocketmags account so you can access the library.



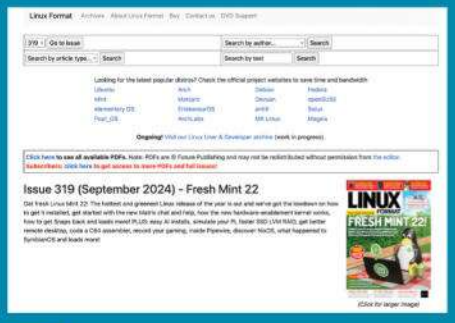
3 Activate your sub

On the Activate A Subscription page, select Linux Format from the pull-down menu and click on Go To Activation Page. Select Library – happy browsing!

THE CLASSIC LXF PDF ARCHIVE

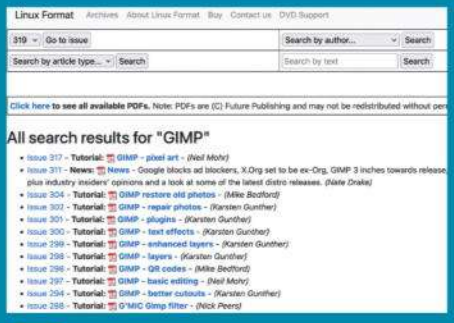
1 Head to Linux Format

You heard the man – browse to **www.linuxformat.com/archives** using your favourite web browser. Where it says Subscribers: Click Here, do just that.



2 Use your subs number

Log in using your subscriber number and surname. You can find the number above your name and address on the address label.



3 Access your issues

Use the issue drop-down menu to access past and current issues. You can also search by author and keyword or article types.

Visit **www.magazinesdirect.com/linux-format**

Framework 13

Brandon Hill loves what this is doing for his eyes!

SPECS

CPU: Intel Core Ultra 7 155H
GPU: Intel Arc Graphics
Mem: 32GB A-Data DDR5-5600 (2x 16GB)
SSD: 1TB WD Black SN850X
Screen: 13.5-inch, 3:2, 2,880x1,920
Comms: Intel Wi-Fi 6E AX210, Bluetooth 5.2
Ports: 3.5mm jack, four user slots
Camera: 1080p, privacy switch
Battery: 61WHr
Power adapter: 60W GaN charger
OS self-install: Fedora 40, Ubuntu 22/24.04 official support
Size: 296.63x228.98 x15.85mm, 1.3kg

When it comes to the best ultrabooks, most of us resign ourselves to the reality that the path for future upgrades is minimal. However, Framework takes a vastly different approach, enabling you to replace just about anything inside (and outside) to your heart's desire. The latest take on that formula is the Framework Laptop 13, of which we received the DIY Edition to review. In true do-it-yourself fashion, you need to install the RAM, storage and operating system. Luckily, Framework makes it a painless process and even provides the necessary tools. This time around, Framework has also introduced a new 2,880x1,920 display option with a 120Hz refresh rate, which is a big improvement over previously available displays.

The Framework Laptop 13 remains on an island as the most easily repairable and upgradable laptop on the market, but the £1,700-plus (though it's easy to cut this down to £1,200) asking price of our review unit might mean it has a narrow target audience.

Given the modular nature of the Framework Laptop 13 and the need to be backward- and forward-compatible with replaceable components, its design looks nearly identical to the version reviewed earlier this year (**LXF313**) and goes back as far as 2021. One noticeable difference, however, is with the display.

Torx sense

There are five captive Torx screws on the bottom of the chassis, which must be loosened to access the internals. Let's first commend Framework for the use of captive screws, as they prevent the headache of having to keep track of tiny screws that end up rolling around and can easily get lost. With the Torx screws loosened, you can flip the laptop over and remove the entire top cover, which is held in place with magnets.

The cover is attached to the motherboard with a ribbon cable, but for easier access to the components, we removed the ribbon cable from the motherboard, which only requires a firm tug of a pull loop at the base.

All the usual suspects are replaceable, including the memory (via two SO-DIMM slots), M.2 PCIe SSD, Wi-Fi module and battery. If you need assistance performing these upgrades, Framework provides a QR code beside each component that leads to an explanatory video.

But the upgrades don't stop there. You can replace the internal speakers, touchpad and even the keyboard (available in multiple colours). Want to replace the motherboard, complete with a CPU upgrade? You can do that; a motherboard, including a Core Ultra 5 125H processor, costs £449 directly from Framework.

The Framework Laptop 13 uses a 61WHr battery, and we put it through our endurance test, involving web browsing, video streaming and graphical tests with the screen brightness set to 150 nits. It lasted nine hours and 59 minutes. The Dell XPS 13 doubles the Framework Laptop 13's endurance, lasting 19 hours and 31 minutes. Last year's Framework also lasted longer at 11 hours 38 minutes with the same battery.

The higher-resolution display is great but it does sap battery power.

During this, at the centre of the keyboard, between the G and H keys, we measured 33.3°C. However, the hottest portion of the laptop was on the bottom, where we recorded 39.2°C. These temperatures are much cooler than those we witnessed with last year's model.

The Framework Laptop 13 continues to be a marvel in the laptop world, thanks to its modular expansion cards and the ability to upgrade nearly every major component. Nothing is off limits for upgrades or repairs; you can swap out the motherboard, display, keyboard, touchpad, webcam, memory, Wi-Fi card and even speakers using easy-to-follow video guides.

Overall productivity performance hasn't changed much compared to last year's model since shifting to the Core 7 Ultra 165H. The biggest difference is the inclusion of the 2,880x1,920 display panel option with a 120Hz refresh rate. The increased screen real estate is a boon for increased productivity, but as we saw in the battery numbers, endurance also takes a hit. LXF

VERDICT

DEVELOPER: Framework
WEB: https://frame.work
PRICE: £900 base (£1,700 tested model)

FEATURES	9/10	EASE OF USE	8/10
PERFORMANCE	7/10	VALUE	8/10

Framework continues to be the only game in town for true upgrades and modularity in the laptop world.

» Rating 8/10

4MLinux 46.0

Nate Drake dives into this innovative, lightweight Polish distro to pose the burning question: are you 4M or against 'em?

IN BRIEF

If you need a zippy OS for an old PC, 4MLinux is ideal. We don't recommend it as a daily driver, though, due to settings bugs, not to mention minimal OS security.

SPECS

CPU: 3GHz
Mem: 450MB
HDD: 8GB
Builds: x86_64

The main website describes this distro as "small, independent and general-purpose". The name derives from the supposed four computing Ms: maintenance, multimedia, miniserver and mystery (games).

We encountered a small hurdle when trying to download the OS, as apparently site visitors must make a PayPal donation before being provided with the download link.

Luckily, DistroWatch pointed us towards Sourceforge (**https://sourceforge.net/projects/linux4m**) where we downloaded the ISO.

The download page did inform us, however, that 4MLinux (aka 4M) supports live booting and installing to a hard drive, and supposedly requires 1,650MB and 128MB of RAM respectively.

The menu contains the **4 M** categories, although some subcategories are curiously worded. **C**heck out **M**ystery > **O**ld**C**ool for retro classics.

On booting the OS for the first time, we noticed the desktop, a customised version of JWM, feh and PCManFM. It makes use of Conky to display system stats, where we discovered that the real requirements were somewhat different. During our live boot, the system used around 2.5GB of RAM in our VM.

Still, 4M is extremely responsive; clicking the over-large home icon launched the folder in seconds and the same was true for menus. These deserve special mention, as although regular categories like Internet are present, those four Ms are also listed. For example, the Multimedia section contains programs such as *MPlayer, Asunder* and *GNU Paint.* Our only complaint is that multimedia apps are grouped into *Jackanory*-style subcategories like Let's Watch and Let's Paint.

Special mention should go to the Mystery section, too, which contains a number of retro-gaming greats in the OldCool category, including *Abuse* and *Doom,* as well as Linux classics such as *Neko* and *Penguins.*

On booting into the live desktop, we were asked to specify keyboard language and a root password. The text prompt also asks if you live in Europe, but if the answer is no, you can't change your time zone.

Eager to proceed with install, we launched the setup assistant. This uses a text interface but is extremely simple, in that it lists the available disks and just asks you to type the name of the install location (in our case **sda2**). It also installs the LILO bootloader automatically. We couldn't see any options for system encryption, nor for creating a dedicated user account. Nevertheless, this was one of the fastest setups we've ever recorded, with files copied to the hard drive in under 10 seconds.

Upon reboot, we were asked once again to choose a keyboard layout and root password. While the system didn't live up to the website's claims of using just 128MB of RAM, the roughly 450MB it did consume means 4M will likely run even on legacy hardware. We were also pleased to see that the install footprint was under 8GB, despite the number of pre-installed apps.

As the default *NetSurf* browser seemed rather unremarkable, we fired up the Applications menu to install more programs. This is managed by the Extensions section, with categories like NetApps or Office. Clicking on an app launches the terminal, where you need only tap `y` to proceed with install. We used this to download and automatically launch *Firefox* in eight seconds. This is seemingly the only way to install additional programs, as 4M has no package manager.

Unfortunately, we started encountering bugs when going into system settings. Although we were able to change the default resolution easily enough via the drop-down menu, this caused the OS to crash. When we attempted to do the same manually using `xnrandr -s`, the 4M dock vanished altogether.

VERDICT

DEVELOPER: Zbigniew Konojacki
WEB: https://4mlinux.com
LICENCE: GPL v3

FEATURES	7/10	EASE OF USE	8/10
PERFORMANCE	9/10	DOCUMENTATION	7/10

4MLinux is lightweight and fast, with lots of apps. **S**till, we'd like to see system encryption and a package manager.

» Rating 8/10

Liya 2.0

Nate Drake takes a gander at this rolling Arch-based distro offering powerful performance and a slick interface. Shame about the updates.

IN BRIEF

During our tests, Liya failed to update properly so we can't recommend it as a daily driver. Nevertheless, setup and installing extra apps worked flawlessly. It's also light on resources.

SPECS

CPU: 2GHz
Mem: 4GB
HDD: 15GB
Builds: x86_64

Based on Arch, Liya follows a rolling release model. Most of the development is undertaken by just one person who takes time on the website to explain who Liya is not for.

Firstly, the OS is only available for 64-bit machines. Secondly, he cautions on the user forum that Liya is not for "distro hoppers" or those who dislike *Systemd*.

The latest version (2.0) is available with either the Cinnamon or Mate desktop environments. We downloaded the hefty 3.6GB ISO of the former and fired it up in a VM.

We were cautious about doing this, as from reading other experiences online it seems the version of Liya released last year threw up a number of PGP key signature errors during setup. The latest release, however, loaded the desktop in under 12 seconds. A number of windows appeared on top of each other on the first boot of the live environment, chief of which was the *Emote* emoji picker.

After dismissing this, we were able to view the *Calamares* installer to proceed with setup. When choosing the install partition, the default filesystem is set to Btrfs, though you can change this to others, such as ext4, if you wish. System encryption is supported.

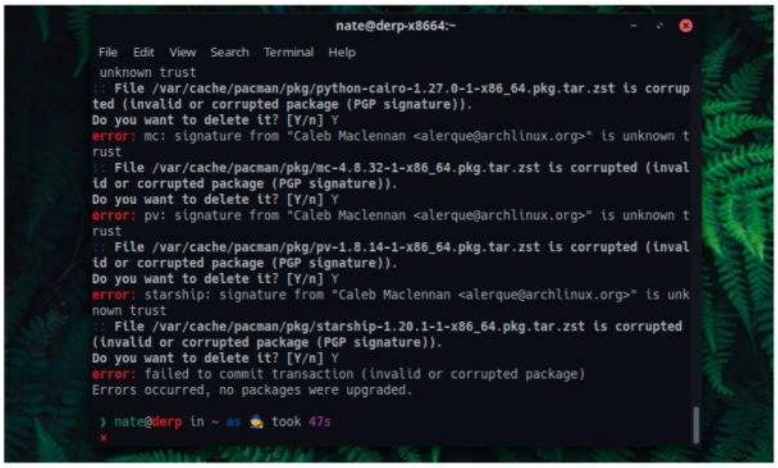
After clicking Install, the splash screen stated that setup could take 15-60 minutes but it completed in under eight. Upon restarting, we decided to visit the user forum (a shortcut to which is provided on the desktop) and follow the official advice to run a system update post-install.

After attempting to refresh the mirror list by the command line, as recommended, the download timed out. Attempting to refresh the keyring raised PGP keyring issues like those outlined above. We then tried to run `sudo pacman -Syu` to execute the update itself.

As this was running in the background, we took the opportunity to launch *Task Manager* to see how well Liya tallied with the stated system requirements. CPU usage remained around 1.5GHz during the update process. The OS also used only around 840MB of RAM, well below the recommended 4GB.

Unfortunately, the update stalled, once again due to errors validating PGP keys. Undeterred, we decided to check if the same issues occurred when downloading new software, so launched *Add/Remove Software*. We tried to install *Firefox* alongside the default *Brave* browser only to encounter a PGP signature error again.

After consulting the Arch wiki, we were ultimately able to fix this issue by running `sudo pacman-key --refresh-keys`, then `sudo fixpacrepo`. After doing so, *Firefox* installed without issue, but we encountered the same issues when trying to update the system via *Pacman*, even after restarting.



During our tests, each time we tried to run a system update via Pacman, Liya encountered PGP key verification errors.

Naturally, we encourage readers who are comfortable with Arch-based systems to contact *Linux Format* if they find a workaround. Still, broken system updates mean that Liya is unsuitable as a daily driver, given potential security issues.

Nevertheless, we found the Cinnamon desktop to be extremely responsive as we navigated menus and launched apps. The rather large ISO is justified by the number of pre-installed programs, such as *OnlyOffice, Celluloid* and *Bleachbit*. We were also gratified to see the OS bundles *Proton VPN* and the aforementioned *Brave* for safer browsing. The careful inclusion of apps also seems in line with the wiki's claim of "No Bloat" when it comes to software.

This is about the extent of the help you'll receive from the online documentation, however, as there is no dedicated troubleshooting section. LXF

VERDICT

DEVELOPER: Sayed M. Hisham
WEB: https://liyalinux.gitlab.io
LICENCE: GPL v3

FEATURES	**6/10**	**EASE OF USE**	**8/10**
PERFORMANCE	**8/10**	**DOCUMENTATION**	**5/10**

Update problems are a major issue but Liya is easy to set up and has a good number of pre-installed apps.

» Rating 7/10

PorteuX 1.6

Nate Drake delves into this Slack-based distro. Does it live up to its claims of being fast, small and portable?

IN BRIEF

PorteuX is clearly very useful if you want to keep your OS on a USB stick and don't mind tinkering with ISOs and save sessions. Otherwise, we suggest a mainstream distro.

SPECS

CPU: 1GHz
Mem: 1GB
HDD: N/A
Builds: x86_64

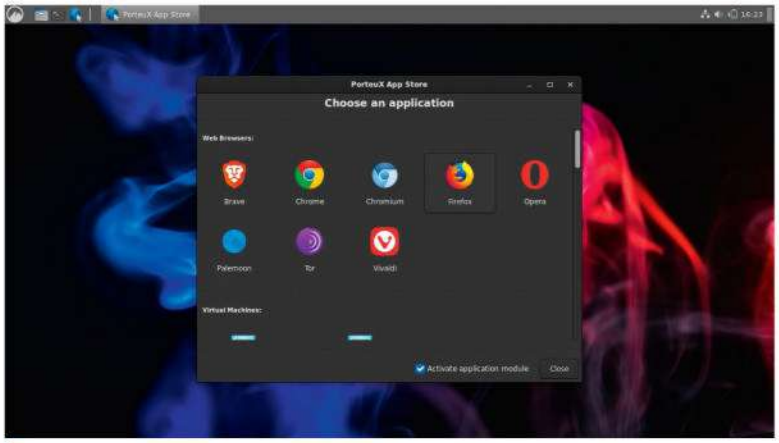
Though not to be confused with Porteus, this similarly-named OS does also make use of modified Linux live scripts.

There's no installer in the traditional sense. It's a modular system, so you can just copy the ISO content to your media storage and run `porteux-installer-for-linux.run` to make the drive bootable.

Presumably, the main use case scenario is someone who doesn't have their own device but wants to use others, in a library, say.

Although there is no dedicated website, the GitHub page has a fairly comprehensive Readme, which also points users to Porteus documentation to familiarise themselves with the concept of these kinds of distros.

ISOs are available for both the current and stable version, with virtually every desktop environment, such as Cinnamon, Mate, Gnome, KDE, LXQt and Xfce.



PorteuX has no default browser, so you need to install one via the app store. Scroll down for other software.

ISO metrics

The GitHub page is keen to point out the relatively small size of the ISOs (the Cinnamon version we downloaded was only around 500MB) and that they typically only require around 1GB of RAM.

While we would have preferred more detailed requirements, upon firing up the ISO, the system monitor informed us that at rest PorteuX's Cinnamon desktop only used around 800MB of RAM.

The documentation had also informed us that PorteuX comes with no default web browser but upon launching the default app store, users are presented with a number of choices. After choosing *Firefox, App Store* also prompted us to choose between stable and ESR releases, as well as to specify the language.

After being told the application module was being processed, the browser installed without issue. Upon relaunching *App Store,* we saw that users can, in fact, scroll down to install other categories of software, such as Virtual Machines, Drivers and Games.

This wasn't immediately obvious upon launching the application and a clearer category section would be more helpful. Still, a visit to *App Store* is necessary given the minimal number of pre-installed programs. Apart from not having a browser, PorteuX also has no office suite and media playback is handled by *MPV.* We could, however, install *LibreOffice* without issue.

As the OS isn't installed to a drive in the traditional sense, programs would need to be redownloaded if the ISO is booted in live mode. PorteuX resolves this via the *Save Session* application. This allows you to save your session, complete with changes, to a single module, file or folder that can load next time you boot.

Crucially, if you've already saved your session, you can also save to an existing file. In our case, we choose to create a new file, which opens a helpful wizard. From here, we were able to specify a savefile name, size and location. There's also an option to encrypt the savefile, the password for which is required on boot.

Once your session has been saved, the application also opens the config file, which is where you can edit the APPEND line to specify the location of the savefile. We weren't able to copy and paste this from the dialog that appeared. Frankly, we felt this could have been done automatically without user input.

PorteuX's other custom utilities are much more intuitive. *Language Switcher* offered a helpful drop-down menu, although if you want any language besides US English, you need to download a specialist multilanguage pack.

PorteuX also boasts its own firewall, which is disabled by default. Upon launching there are buttons for common functions like Strict or Normal rules. LXF

VERDICT

DEVELOPER: Porteux Core Team
WEB: https://github.com/porteux/porteux/
LICENCE: Mainly GPL

FEATURES	6/10	EASE OF USE	7/10
PERFORMANCE	7/10	DOCUMENTATION	3/10

Offers a variety of desktops and smooth performance. Saving sessions is intuitive but probably not for beginners.

» Rating 6/10

RebeccaBlackOS 2024

Nate Drake is getting that Friday feeling as he discovers all that's awesome about this celebrity-themed distro with Wayland support.

IN BRIEF

Rebecca BlackOS stands alone in booting Wayland from live media. It offers easy setup and a range of desktop environments. We've deducted points due to issues with Weston and installing software.

SPECS

CPU: 1GHz
Mem: 1GB
HDD: 8GB
Builds: x86_64

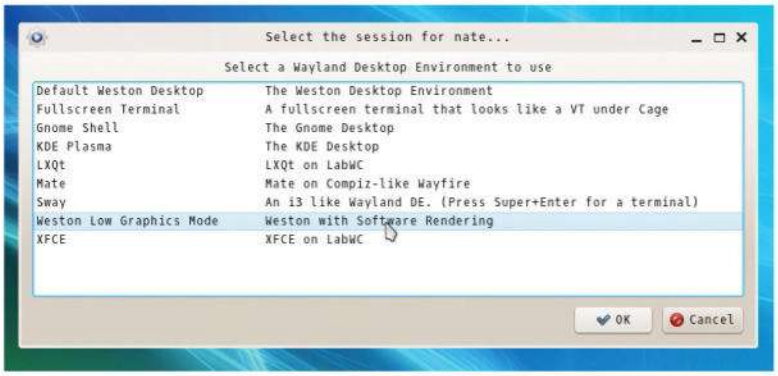
Celebrity-themed Linux distros are nothing new. Who can forget the likes of Biebian – a mishmash of Debian with a Justin Bieber theme, or even Hannah Montana Linux?

The developer, who's wisely decided to remain anonymous, is a fan of the eponymous celebrity and has been developing this Debian-based OS since 2016. As with PorteuX, they clearly felt a dedicated website was beneath them but they have uploaded an extensive Readme to Sourceforge.

It was here we discovered that the latest version is for 64-bit systems only, due to *"QtWebEngine* refusing to build in 32-bit chroots". The developer also cautions that the OS does not contain any of Rebecca Black's images or music, encouraging users to purchase these legally instead. Trying to conceal our disappointment, we downloaded and booted the 1.5GB ISO in a virtual machine.

Once the OS loads, you are presented with a choice of desktop environments. The default is Weston, given that this distro only supports Wayland sessions. In fact, according to DistroWatch, RebeccaBlackOS is the only known version of Linux to do this from live media.

You can also choose a more familiar desktop environment, such as Gnome, Xfce, LXQt, Sway, Mate or KDE, or can choose to boot to a full-screen terminal.



On boot, you can choose from a variety of desktop environments. When using Weston, however, we had issues changing the resolution.

Nice and breezy

After choosing the default desktop, we launched the system installer, which seems to be based on *Calamares.* This meant setup was a breeze, only requiring simple steps like choosing the language and installation partition. System encryption is supported.

The install process itself took just over three minutes in our virtual machine. Upon restarting, we found a login dialog appeared, whereupon users must click on the username they created during setup and enter the password.

Unfortunately, upon logging in to the default Westland session, we encountered an error when trying to change the display resolution. We attempted to switch to LXQt and received a notification that the desktop was attempting to start but it failed to load.

Fortunately, Gnome Shell resolved without any issue and allowed us to rejig the display resolution. We also took the opportunity to run the bundled *glmark* utility, which is designed to measure OpenGL system performance.

Next, we fired up *Weston Terminal (Konsole* is also available) to run `echo $XDG_SESSION_TYPE` to check Wayland was running. We also took the time to run an update, which is where we discovered that the latest version of RebeccaBlackOS is based on Debian 12 (Bookworm).

As the distro has no official website, we also checked *Gnome System Monitor* and discovered the OS was using around 1.3GB of RAM and 68% of our virtual 1GHz CPU. The install footprint was also reasonably light at just under 8GB.

The latest RebeccaBlackOS has had X11 removed entirely from the startup path. However, when we ran the *GTK VTE Terminal,* we saw it's possible to force programs to use their X11 back-ends via the command `xwaylandapp`. We used this to launch the pre-installed *Elisa* music player without issue.

The terminal also allows running Wayland programs as root via `wlsudo`, which we used to launch the default *Konqueror* browser.

Although *Gnome Software* was noticeably absent, in theory you can add further programs using *KDE Discover.* During our tests, however, *Discover* was unable to load available apps and froze during updates. We were, however, able to install programs from Debian repositories via the terminal with `apt install`. LXF

VERDICT

DEVELOPER: n3rdopolis
WEB: https://sourceforge.net/projects/rebeccablackos/ **LICENCE:** Mainly GPLv2+

FEATURES	**7/10**	**EASE OF USE**	**7/10**
PERFORMANCE	**8/10**	**DOCUMENTATION**	**2/10**

Comes with Wayland support and a huge bundle of pre-installed apps. Sadly, it's a little buggy for everyday use.

» Rating 6/10

World of Goo 2

Recycling goo has The Management ecstatic at the chances it'll give **Kerry Brunskill** to reuse all of their old work to increase profits.

SPECS
OS: Ubuntu 24.04
CPU: Intel Core i5 dual-core 2GHz
Mem: 8GB
GPU: Intel Iris
HDD: 2GB

It's been 16 long years since *World of Goo* squidged its way into our hearts and hard drives. The much-loved physics-based puzzle game was one of modern indie gaming's earliest and biggest successes. But a lot has changed since then.

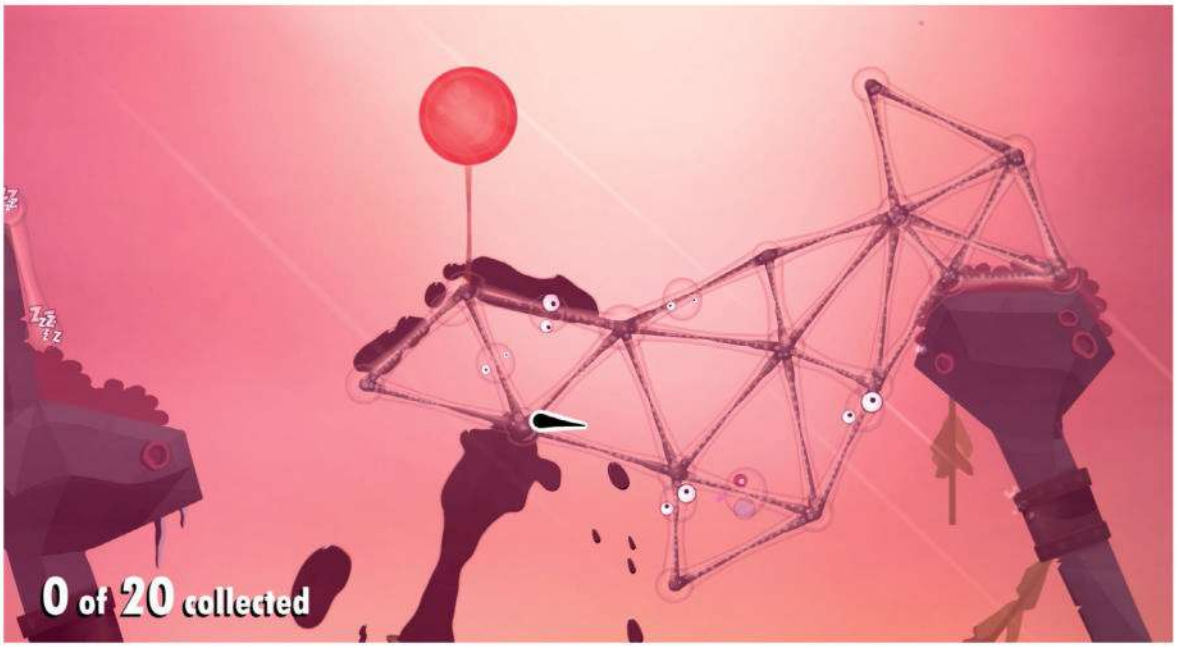
Polished games from unfamiliar studios with barely a dozen staff members aren't some headline-grabbing curiosity any more; they're absolutely everywhere, all the time. On our PCs. Our Steam Decks. Our phones. They're often discounted and bundled up, and at times even given away for free. Who needs more *World of Goo*, then, when the original was lovely and creative, and very much done and dusted years ago?

Playing this game is a lot like catching up with an old friend. It looks, feels and plays in much the same way it did before. Everything from the artistic stage select design to more than a few goo types, environmental hazards and puzzle pieces call back to the original gooey head-scratcher. Even the little time-rewinding bugs have returned (thank goodness), making it easy to either try out a few wild ideas or finally perfect a tough segment. Within moments, it was as though not only had the game never been away but we'd never stopped playing it either.

And because of that, clearing the first few challenges wasn't just easy – it almost felt instinctive. There we were, watching towers of goo dangerously sway as we grabbed, built and stretched an increasingly wobbly mass towards the exit pipe that would suck it all up and end the level, same as last time. Just like the ancient past of 2008, sometimes we needed to hook balloons on to a goo lattice to move it around, flipping it over and over to 'walk' it across the landscape or encourage it to roll in a particular direction. At other times we needed to set something on fire or watch out for draughts. There are a lot of obviously recycled ideas in here.

These old favourites don't exist in an innovation-free vacuum, but are mixed in with high-speed train rides, little boating expeditions, tense journeys through the

It's goo, Jim, but not as we know it.



The classic gameplay is back but with new twists and drips to contend with.

dark, and even a spot of goo-themed golf. So when an old concept makes a comeback, it feels like the welcome return of a good idea, a greatest hits compilation of some of the finest puzzling PC gaming has ever seen. *Goo 2* isn't rehashing old favourites because it's out of ideas (some of the new goo types lead to some incredible and unsettling puzzle solutions – we'll never forget slowly forcing a giant goopy mass through some painful spikes using expanding goo balls), it just wants to show us a good time, while offering a few amusing signposts and cutscenes filled with pointed comments on corporate greenwashing, commercial-minded environmentalism, and the cynicism of sustainable capitalism along the way.

This good time is elevated by generous flexibility. The goo is as pleasantly tactile as ever, a fun substance to play with no matter how well you're doing. It's always satisfying just to imagine something and then try to build it, even if the dangling mound of goo balls you end up with isn't the most efficient use of anyone's time and resources. Tough challenges for each stage are always available – and optional. It's up to you if you ignore them completely, come back to some of them later, or refuse to move on until they've all been cleared in a single run.

We didn't even have to clear most of the stages at all unless we wanted to, as most levels are only a couple of clicks away from being skipped without judgement. Maybe we didn't fancy this puzzle right now. Maybe this particular area's puzzle theme didn't do anything for us. Maybe we just want to open everything up right away and then dive into whatever we feel like taking on today, happily saving the rest for later like the treats they are. Being able to mostly choose what we played and when helped to smooth the frustration and restarts that can come with something as potentially woolly and unpredictable as a physics-based puzzler. Did we find one level to be a little too highly tuned for our tastes or current lack of patience? Then we didn't have to play it.

Unfortunately, we couldn't ever escape the odd background colour choices used for many areas. Lots of places used a light hazy wash of something non-specific as their backdrop, which looks great in screenshots but meant the helpful white lines used to indicate how the floating balls of goo connect were often hard to make out. In most levels, this was an annoying lack of contrast, and meant it sometimes took longer than necessary to build a solid bridge of goo. In others, it was a challenge-run-ending issue, and something that surely should've been picked up before we got anywhere near the game.

But this was honestly the worst thing we found in *World of Goo 2*, and even when it did cause us problems, we were generally having too much fun to really mind the odd stumble. It's a clever, surprising game that celebrates all the goo that came before and all the goo here now. It's goo[d] to be back. LXF

Even fire can't seem to beat the goo.

VERDICT

DEVELOPER: 2DBoy
WEB: https://worldofgoo2.com
PRICE: £24

GRAPHICS	7/10	GAMEPLAY	8/10
LONGEVITY	7/10	VALUE	7/10

Familiar and inventive, tough yet easygoing, **World of Goo 2** is a whole world of fun.

» Rating 8/10

WE COMPARE TONS OF STUFF SO YOU DON'T HAVE TO!

Roundup

Kodi » Universal Media Server » Jellyfin » Plex Media Server » MiniDLNA

Michael Reed is a consumer of media, both offline and on. He'll consume it remotely or locally if he has to.

Media servers

Without wishing to turn into a couch potato, **Michael Reed** examines five media servers that mean he could become one if he decided to.

HOW WE TESTED...

We used a fresh installation of standard Ubuntu (24.04) as our test platform for all the servers. One concession to this role we made was to choose the Basic install in Ubuntu's installer as we didn't have a need for extra apps. We also added and configured an SSH server so that we could handle some of the configuration details remotely.

We added Flatpak to Ubuntu's existing Snap support to broaden our options when going after the latest stable versions of server applications. Wherever we could, we favoured this simplified method of installation rather than doing full manual installation of each candidate. No point in making things difficult if they don't have to be.

Where we were able, we did the setup through a combination of remote access to the web interface of the candidate with some SSH access when needed, rather than hands-on with the server itself.

This month, we're looking at solutions that enable you to build a Linux-based setup for interacting with your media library of TV shows, movies and music. Some of these servers directly output to an attached display, and some of them work over the network. Some can do a bit of both. Beyond that, some of them can even be expanded beyond the basic functions of playing back media by adding things such as gaming and web browsing features.

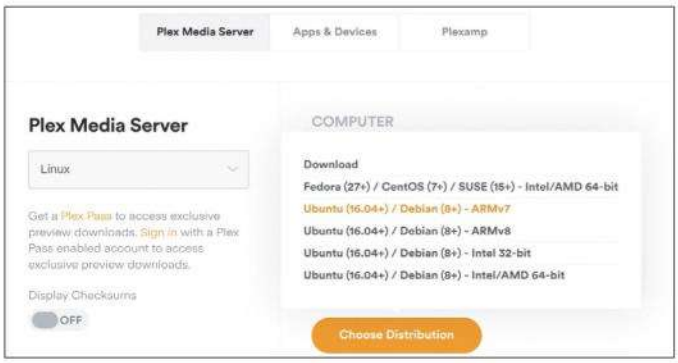
You might prefer to dedicate a computer to server duties, but you can run each of these servers on your main computer or within a virtual machine if you wish. Your media can be on the server, or you can point the server towards a storage location on your network.

Once the server is up and running, it's time to consider the client that you use to access your content. For example, you may well wish to install a client app on your TV or your mobile phone. All of the media servers we're looking at can also transmit content via open standards, meaning that you can access that content through standard media players without having to set up anything special software-wise – handy in a pinch.

CREDIT: Fotolia/saniphoto 2008

Basic server install options

Hopefully, you've got a good array of options for setting up the server.



Although largely a proprietary system, Plex has packages for the most common Linux distros. But you could get stuck if trying to run it under a more obscure setup.

Plex Media Server interacts with the *Plex* website in use, but runs independently on your system. It's largely proprietary. It was absent from Flathub and the Snap version was out of date, so we downloaded the DEB file offered by the *Plex* website. The website offers various native Linux packages in RPM and DEB format, 32 and 64-bit PCs, and packages for Arm-based systems such as the Raspberry Pi.

Kodi is a client with some server features and connects to a display rather than being completely network bound. It's open source, and the website recommends Flatpak installation, so that's what we did. There are manual options, too, particularly on Debian-based OSes. Most popular platforms are supported.

Jellyfin is open source. The *Jellyfin* Flathub package was out of date and we couldn't find it on Snap, so we installed the stable package from the website. Most major Linux distros are covered with specific packages and there is a generic Linux **tar.gz** archive. We cut and pasted a command line from the website that downloaded everything we needed for Ubuntu installation. There are also Arm packages for the likes of the Raspberry Pi.

MiniDLNA is open source. It hasn't been updated in a while, so we installed it using Ubuntu's package management tools, and most distros have it in the repos. Given the command line and configuration file workflow of *MiniDLNA*, we preferred this as it means the configuration files are placed in standard locations.

Universal Media Server is open source, but the official builds offered on the website are one version behind those available to Patreon sponsors. Building from source is possible, but the build script failed when we followed the instructions, so we used the slightly older official package from the main website.

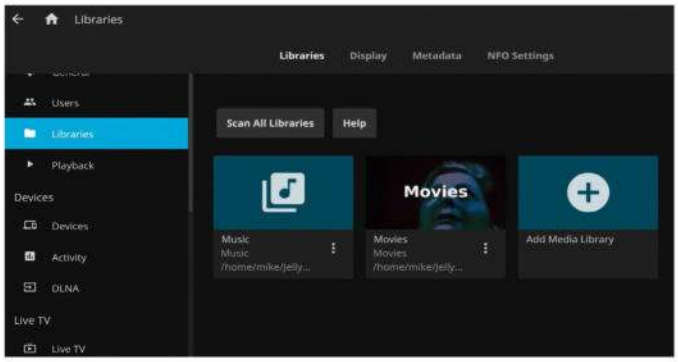
VERDICT

KODI	10/10	PLEX MEDIA SERVER	7/10
UNIVERSAL MEDIA SERVER	6/10	MINIDLNA	9/10
JELLYFIN	7/10		

Projects such as Kodi and Jellyfin are featureful but completely open source, giving maximum flexibility when it comes to installation options.

Initial setup process

Let's get some folders set up, media shared and cover other basics.



The Jellyfin interface is browser-based and divided into categories accessed via the sidebar. It's an efficient way of working, and it can be accessed remotely.

Jellyfin's configuration is carried out via the web interface. We faced a little bit of confusion when adding media folders, as the scrolling list of options contained entries that made no sense (such as **/usr/lib/x86_64-linux-gnu/openh264**). In most cases, you have to manually add the path, which can be awkward if you're accessing remotely.

Plex Media Server also has a web-based configuration layout that is similarly quick to navigate, well organised and full of options.

Universal Media Server uses a Java application for its configuration, but it also offers web-based configuration, giving the best of both worlds.

MiniDLNA's configuration file is well commented and it's a good idea to go over it to set up a few details, such as the directories that contain the media you want to access. However, the server does need to be restarted after changes are made. The command line tool, *minidlnad*, can force rescans and so on, but the server sometimes needs to be shut down for this to work. Some Linux users appreciate a command line and text file configuration like this, even if it's a bit fiddly in places.

Generally, the *Kodi* configuration is carried out via its own user interface running on the computer that *Kodi* is running on. It's an attractive interface that can even be navigated via a remote, but some things, such as cutting and pasting, can be awkward. It feels like the settings page in a computer game, and we reckon that the average user would be faster navigating through a multi-page, categorised web interface.

VERDICT

KODI	7/10	PLEX MEDIA SERVER	8/10
UNIVERSAL MEDIA SERVER	8/10	MINIDLNA	7/10
JELLYFIN	8/10		

Configuration files or an on-screen UI are good, but we think that the web-based settings pages hit the efficiency sweet spot.

Network access to content

Open standards over the network.

There are two main standards for accessing a library of media content: DLNA and HTTP. DLNA allows the library to be browsed with a client such as *VLC*. HTTP means that you can navigate the content via a web browser, and this tends to offer a fancier user interface, with individual graphics for each piece of content.

Having a separate server enables a setup with a low-power device, such as a Raspberry Pi. That way, the server can be located out of the way and will still work if your main computer is ever out of action or simply switched off.

Most of these servers only use a small part of your total CPU and memory resources if you add them as a piece of software. This might be more power-efficient than running a separate computer, and if you need more resources, adding more memory and a faster processor to your main rig might be more cost-effective than upgrading a server.

Kodi 7/10

Kodi is designed to be plugged into a display and operated via a remote control but it can also share content over the network using the DLNA and UPnP standards. What you'll end up with depends on the DLNA client that you are using, but it tends to be a bare-boned list of content within a set of pseudo directories (such as **recently added**).

Kodi isn't designed to be wholly operated over a web connection, but it does have an HTTP-based web remote-control feature. Unfortunately, this doesn't stream media into the browser, it just allows you to control your *Kodi* box via a web browser. This remote control could be used on a tablet, phone or even a laptop, and it features the same high-quality metadata as the main *Kodi* interface. It's an intriguing prospect to have a remote control that features box art, summaries, reviews and ratings when selecting between different content.

Universal Media Server 8/10

Universal Media Server is all about access over the network and it offers both DLNA and HTTP access to your media. The Java-based front-end is great in this respect as it immediately updates when it finds a DLNA client on the network and gives you a graphical overview of all such devices. Oddly, the fine-tuning DLNA and HTTP options are made by editing config files directly, and those are launched through the GUI. It's inconsistent, but at least the options are available.

Universal Media Server has a web-based browser/player. Like the *Jellyfin* HTTP interface, it's bare-boned, but it's quick, and it has all the expected features for accessing a collection. Among the usual meta-folders there is access for transcoded versions of your files, which is useful. It's an interface that isn't designed to be used on a large screen with a remote control, but DLNA does work on smart televisions with a suitable app.

Access over the internet

For when you want to access your media wherever you are.

Most of these servers were disappointing when it came to getting them working over the internet and it would require a bit of know-how to get it working securely.

MiniDLNA uses DLNA only. The problem is that DLNA is designed to be used on a local network rather than over the internet. As with all servers of this type, you could maybe set up some sort of secure remote access, but you'd have to do the work of setting it all up yourself.

Kodi is, effectively, a client application even though it can serve content over DLNA. What you could do is set up *Kodi* on the remote platform and then point it to the same network storage resources that your home setup is accessing (using secure protocols). There is also a settings backup add-on that can work between different setups. This would give you a reasonably consistent experience between devices.

Jellyfin's HTTP web interface also has an HTTPS mode. This requires a bit of work on the part of the user in terms of forwarding the traffic and providing a valid certificate.

Universal Media Server isn't actually designed to operate over the internet. As with anything that offers HTTP connectivity, it's fairly easy to forward ports on the router and get it working, but we'd recommend against doing that for security reasons.

Plex is the clear winner here because it's designed with internet access in mind. This means that you can get to your server or access it via the *Plex* website or official apps at all times. The security and set up work is done for you by the *Plex* system.

VERDICT

KODI	5/10	PLEX MEDIA SERVER	10/10
UNIVERSAL MEDIA SERVER	4/10	MINIDLNA	4/10
JELLYFIN	7/10		

With no real effort on the part of the user, Plex is automatically internet-enabled. The others require work.

Jellyfin 8/10

Like the other clients, *Jellyfin* can operate as a DLNA server. There are options in the preferences that could be helpful when troubleshooting the DLNA connection, including the welcome sight of full, verbose logging. *Jellyfin* can also play to a DLNA-enabled device, allowing the server to control an appropriate device. There is advice within the configuration pages that you can use settings profiles to make only certain libraries available to certain users.

Content can be browsed and played back over HTTP. It's similar to *Universal Media Server* in that it's a plain interface that gets the job done rather than one that's heavily skinned and animated. It's functional even though it's unlikely to elicit any wows on aesthetic grounds. As with all of these types of setup, it's not particularly useful for use on a TV; that's what the official client apps are for. HTTPS support can be enabled for secure use over the internet but it requires extra setup work.

Plex Media Server 8/10

As well as being accessible from official client apps, *Plex Media Server* is a fully-featured DLNA server, with a handful of troubleshooting options in its preferences. This gives access to your personal media files via DLNA if you use a suitable client, but not the ad-supported streaming content the overall *Plex* service offers.

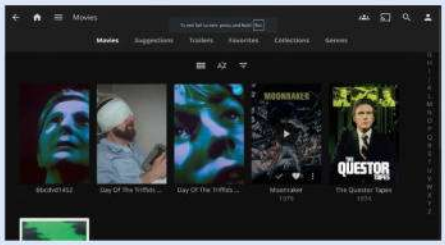
Plex Media Server can also be used over an HTTP connection, which gives a mouse-operated interface similar to the one offered by the client apps, and like most of *Plex*, it looks slick. You can access this service from within your own network, over the internet or via the *Plex* website. The relevant preferences page details the connection settings needed for each approach. *Plex* itself handles all the security measures while you're accessing this content over the internet. HTTP access does give access to both your private media and the ad-supported streaming content.

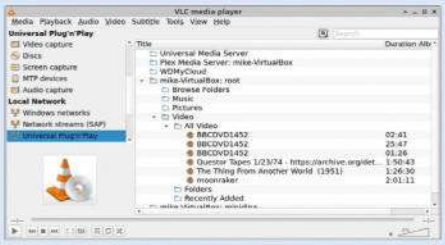
MiniDLNA 6/10

The whole point of *MiniDLNA* is that it offers basic DLNA access to your media library. When you're using a client such as *VLC* to access this content, it doesn't make much difference if you have an elaborate piece of software with loads of features serving this content as it all looks the same on the client end.

We recommend enabling inotify so that content is updated as soon as it is added. There is a slight increase in resource usage from doing this, and low resource usage is part of the appeal of *MiniDLNA*, but it's not a huge issue.

MiniDLNA serves content in a basic and reliable way, but one snag with the way that it works is that it doesn't transcode at all, meaning that your chosen player must understand the codec of the original file. So, for example, a device such as a PlayStation 3 might not be able to understand a newer or obscure file format.





Support and documentation

Help to get it set up, make it work how you want and troubleshoot issues.

Universal Media Server is reasonably well known around the internet, but we were underwhelmed by the amount of activity on the official forum. The knowledge base has some useful articles, but once again we found the documentation scene for *Universal Media Server* to be rather lacking. If you can't get a feature to work, you may end up stuck.

As ever, popularity counts for a lot, and *Kodi* is one of the better-known media servers. For example, when solving a problem, expect to find support threads on sites such as Reddit. Returning to the official support routes, the many subforums are busy, which is what we like to see. Most of the documentation is in the form of articles located in the wiki and they cover every single area of the program. All in all, you can probably accomplish what you want to do using the available support.

The *Jellyfin* documentation takes the form of a series of articles, and it's highly professional and well organised. The forum is perfectly active and it was clear that questions and discussion points were being responded to.

Due to the largely proprietary nature of *Plex*, it's not surprising that the support information was a bit hidden on the website, but when we did locate it, we couldn't fault the collection of support articles. There's about 100 of these and they cover every area of the software.

It's a fairly simple program, so it's forgivable that most of the documentation for *MiniDLNA* is fragmented around the internet. In our case, we used the guide on the Ubuntu site to set it up.

VERDICT

KODI	10/10	PLEX MEDIA SERVER	8/10
UNIVERSAL MEDIA SERVER	5/10	MINIDLNA	6/10
JELLYFIN	7/10		

Kodi's popularity gives it a built-in advantage. The Plex site had a surprising amount of information by proprietary software standards.

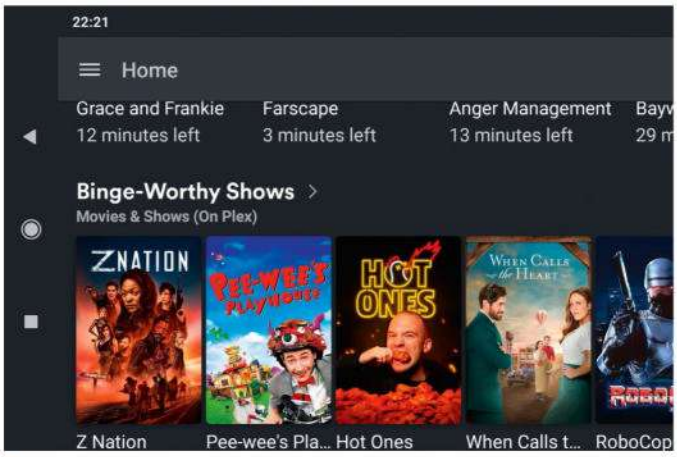
Official client app

The clients are front-ends that allow you to interact with your content.

When you install the *Kodi* app, you are generally installing a full version of *Kodi* itself, and this has some implications. Typically, the user interface is appropriately scaled for a device such as a mobile phone or a television. As it's more than just a simple client, it's a bit heavy on resources. Some aspects of a setup can be copied across, but you largely set up each installation from scratch. On the plus side, nearly every platform you can think of is supported, from Firestick to iPad to desktop PC, even though some platforms may struggle to run it.

Jellyfin has official clients for iOS and Android mobile devices, smart TVs and desktop computers. The interface smartly arranges your TV shows, films and music, and like the HTTP interface, it's functional while lacking visual flair compared to *Kodi* or commercial streaming services such as Netflix. However, it is fast and low on resource usage.

Even though they are usable with a free account, playback of your personal media and some other features require a *Plex* subscription. Pretty much every type of mobile device you can think of, along with single-board computers and smart TVs, are supported as targets and are usually available through official channels without the need for sideloading. As with all of the *Plex* stack, the clients have an attractive, highly professional appearance.



Plex's Android client on a mobile phone. As with all of Plex, presentation is superb.

MiniDLNA and *Universal Media Server* are similar in that they both present their content using DLNA and UPnP, but they have no official clients. There are various clients for televisions, mobile devices and desktop computers that can browse and play this content.

VERDICT

KODI	6/10	PLEX MEDIA SERVER	7/10
UNIVERSAL MEDIA SERVER	2/10	MINIDLNA	2/10
JELLYFIN	7/10		

Plex's client apps are good, but have some restrictions on the free account. Jellyfin's clients are perfectly usable.

Add-ons and extra features

Pushing systems beyond their basic media server functions.

As it's a network-based server, *Universal Media Server* doesn't have a lot of scope for adding utilities. However, you can add streaming video feeds. In particular, the official wiki explains how to add YouTube channels, which present themselves via the interface of the DLNA client you are using.

Kodi has an extensive add-ons ecosystem consisting of official and third-party modules, and these are installed from within the configuration user interface. In fact, the selection is so extensive that it borders on overwhelming. The add-ons consist of smaller utilities such as a weather app and game changers such as support for emulators and other ways to play games. Adding a web browser can be useful,too. It can be fiddly to locate a plugin and configure it, but nearly anything is possible once it's working.

Plex isn't just a conduit to your own private media stash, it's also a significant source of ad-supported films and TV shows, and your own media is mixed in with all of this when using the official app. If you're already watching *Plex*, it might push you in favour of using the *Plex* server for your own media. *Plex* has an add-on system that is largely made up of utilities to improve interoperability with other services.

Kodi has a massive library of add-ons that are downloadable and installable from the interface. These can change the way that Kodi works or add new functions.

MiniDLNA does one thing and does it well, so doesn't have an add-on system. You could say it lends itself to staying out of the way while you add extra facilities through other server software.

Jellyfin has an add-on system with about 25 plugins. Most are orientated towards interacting with other systems and standards rather than adding significant features like the *Kodi* add-ons. It's a decent showing that might enable a convenient way of working.

VERDICT

KODI	10/10	PLEX MEDIA SERVER	6/10
UNIVERSAL MEDIA SERVER	4/10	MINIDLNA	2/10
JELLYFIN	6/10		

Kodi is the king when it comes to adding new features and enabling niche customisations through add-ons.

The verdict

Media servers

The reason we chose *Kodi* as our winner is that it's so versatile. It's hard to think of a way of working with media files that *Kodi* can't be persuaded to do. Often, when we considered the advantages of one of the other servers, we noticed that *Kodi* could work like that as well. It's also the only one of these servers that can be connected to a TV as a true HTPC (home theatre personal computer) without depending on being connected to a network. On the other hand, it can also be used as a network server using the DLNA protocol.

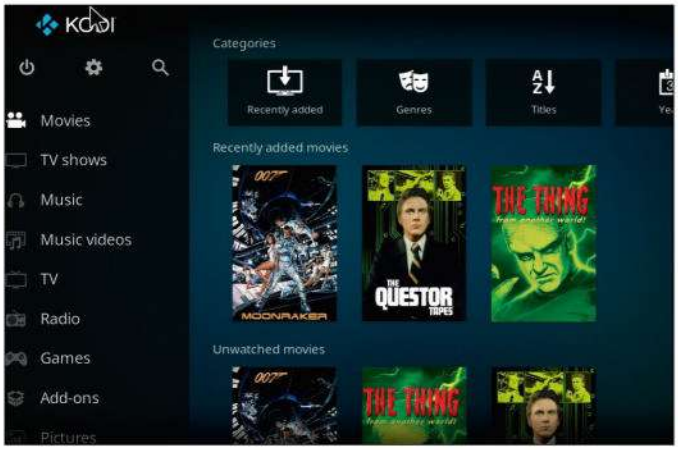
The add-ons system means it can do other things such as working as an emulator front-end. The user interface is extremely attractive, and it's handy being able to access nearly every feature via a Bluetooth remote control. We've a feeling that people who see your *Kodi* box in action are likely to be intrigued and want to try it, too.

Jellyfin is open source through and through, which lends it a natural versatility in how it's deployed. As well as being a DLNA server, it's highly accessible thanks to its custom client apps and HTTP access.

Universal Media Player does a good job at what it's designed for: serving various types of media over the network. It offers the best of both worlds because it can be configured either through the Java desktop application or through the web interface. It also has perfectly usable access to your media collection through an HTTP interface.

Plex Media Server has points against it from the start because it's not open source and it requires a (free) account to operate. You can use the basic functions for free, and in all fairness, the pricing is reasonable. As a personal media server, it works perfectly well and the (ad-supported) world of content that it provides is an additional attraction. A lot of the nicer features are only available if you're a paying customer, but generally, it's a great-looking install-and-go system with a lot of refinement.

MiniDLNA is a simple way of sharing video and audio content over your network using the DLNA and UPnP standards. If you like editing a plaintext configuration file and operating things from the command line, *MiniDLNA* is quick to deploy, and it's sitting in the repository of most Linux distributions.



1st Kodi 9/10

Web: **https://kodi.tv**
Licence: **GPL 2.0 or later** **Version:** **21.1**
Can work as HTPC and has network features. Excellent UI. Expandable.

2nd Jellyfin 8/10

Web: **https://jellyfin.org**
Licence: **GPL 2** **Version:** **10.8.13**
Browser-based configuration. Good basic apps and an HTTP interface.

3rd Universal Media Player 7/10

Web: **www.universalmediaserver.com**
Licence: **GPL 2** **Version:** **14.4.0**
Both application and web-based configuration. Basic HTTP interface.

4th Plex Media Server 7/10

Web: **www.plex.tv**
Licence: **Proprietary** **Version:** **1.40.5.8854-f36c552fd**
Slick-looking and highly functional. Some restrictions with a free account.

5th MiniDLNA 7/10

Web: **https://sourceforge.net/projects/minidlna**
Licence: **GPL 2** **Version:** **1.3.3+dfsg-1build2**
Lightweight, simple DLNA server. Command line and config file operation.

» ALSO CONSIDER

Emby (**https://emby.media**) is a media server with a set of custom clients with an attractive user interface. However, when we had the system installed and running, we quickly discovered that you can't even play your own media content without making a one-off payment or starting a subscription. This is an acceptable business model, but it's not for us, particularly because it's not made obvious that you've got to pay for a basic level of functionality.

LibreELEC (**https://libreelec.tv**/) is a specialised, cut-down Linux distribution with just enough functionality to run *Kodi*. The target selection of platforms are what you might expect from a Linux distribution and include desktop PC and single-board computers such as the Raspberry Pi.

OSMC (**https://osmc.tv**) is another small Linux distribution that offers a ready-made *Kodi* setup. Unfortunately, it's only available for Arm-based computers like the Raspberry Pi.

Build Your LINUX FORTRESS!

Chronically insecure and unreasonable **Jonni Bidwell** needs reasonable operational security. With Qubes, and containment and isolation, he finds a solution.

Threats to your security and privacy are everywhere. Even if you're not an enemy of a nation state or elite hacking crew. Any decent desktop Linux distribution does a lot to protect you from threats and compromise. And it can do more if you spend time tweaking it. But what if you're still concerned and want security baked into everything your distro does?

Well, then you look to Qubes OS. Recommended by such diverse entities as NSA whistleblower and guest of the Russian Federation Edward Snowden and the Freedom of the Press Foundation, you don't need to take our word as to its efficacy. Qubes describes itself with much more understatement, identifying as "a reasonably secure operating system". This modesty comes from Qubes's paranoia. It assumes that everything will get compromised sooner or later, so it runs everything in isolated virtual machines.

Not only that but so-called Application VMs (where most apps in Qubes are run) are rebuilt on each use from a template. So, even if they get infected, the next time they're started, they're fine. You can have Disposable VMs, too, in case you really want to leave no trace. And did we mention it comes with the Whonix Tor gateway ready to go? All this magic happens under the lightweight *Xen* hypervisor. So crack out that spare laptop and prepare to build a digital fortress!

CREDIT: Magictorch

The Qube-ic formula

What on earth is Qubes? How can it stop your traitorous activities being noted by hostile governments?

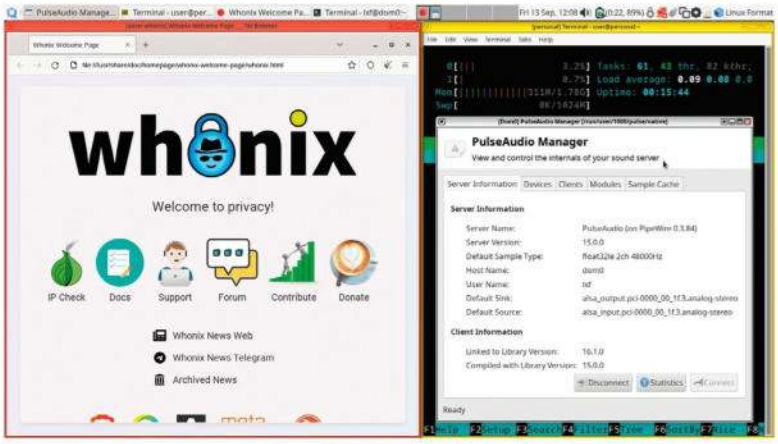
If you're using Linux at home, you're already doing pretty well security-wise. Even reading this magazine has been shown to boost security by up to 15% (in certain circumstances). We can all improve our security, though, but if you've looked into this, you'll know that this generally comes with some sort of usability trade-off. For example, if you really don't want any of your computers and smart devices to be hacked, you could just switch off your router (and any radios in any of those devices). But that's not useful. Qubes OS has a far more reasonable approach, namely security by compartmentalisation. A default Qubes installation sets up four virtual machines (VMs) running under the *Xen* hypervisor. Much like the spoon in the first *Matrix* movie, there isn't (under normal circumstances) a Qubes OS for you to interact with – only these subordinate VMs (known in Qubes as qubes).

One of the qubes (called **Dom0**) is privileged and can access hardware and talk to the hypervisor. But this usually doesn't run some kind bespoke Xen OS. It runs whatever operating system you choose, although for reasons of sanity, most people use some sort of Linux or BSD. The (sensible) default in Qubes is for **Dom0** to run Fedora. The other qubes all communicate through **Dom0** in order to do anything in the real world. In the default config, a single user qube also running Fedora is instantiated, where it's intended you keep your day-to-day, non-secret business. Then there's a qube running the Whonix proxy. There are also a couple of service qubes for managing the display, network, firewall and USB hardware.

It's hip to be qube

You can add as many other qubes as you like, and they can run any OS you like (including Windows). One of Qubes's many impressive features is its ability to spin up Disposable VMs on the fly. These might be used just for a single application, or for several. Either way, no trace of your Disposable qube or activities therein remain after it is shut down. Because it self-destructs. And, like any qube, there's no way for it to interfere with the other qubes while it's running. Qubes has a powerful templating system, so it's easy to modify the default applications included in a given qube. This either saves space or saves time installing critical applications.

Another OS popular with the paranoid and security conscious is Tails (The Amnesiac Incognito Live System). Unlike Tails, which doesn't leave any trace on your storage (unless you explicitly set it up), Qubes remembers any configuration you do to its (non-disposable) qubes. So, in that sense, it's much more like a traditional operating system – your documents and bookmarks survive a reboot. The only immediately obvious deviation from desktop traditions is that windows are colour coded according to which qube they are running in. So, you can keep your various digital lives totally isolated.



Windows in Qubes are colour coded according to which Qube they are running, so you don't get your IDs confused.

» THE XEN PROJECT

The *Xen* hypervisor is open source, otherwise it would be controversial dedicating a whole box to it like this. We tend to focus our virtualisation efforts around Oracle's *VirtualBox* or Red Hat's *Virtual Machine Manager*. This is partly because they're easy to run on your Linux desktop. Such things are commonly called Type 2 hypervisors, which is technically incorrect for reasons you can easily look up. A Type 1 hypervisor, for the purposes of this box, is its own thin operating system (OS) that exists solely for the purpose of running VMs. And that's the kind of animal *Xen* is.

Xen was originally developed at the University of Cambridge and first released in 2003. Since 2013, the Xen Project (so renamed since the Xen name was used commercially in the interim) exists under the aegis of the Linux Foundation as a self-governing, collaborative project (with members including AMD, Citrix, Google, Intel and Samsung). Besides Qubes, *Xen* is used in the XCP-ng OS, (*see* **LXF273**) itself is the successor of XCP (Xen Cloud Platform) – formerly the free version of Citrix's closed source Xen Server). XCP_ng bills itself as a turnkey solution for orchestrating VMs, featuring integration with cloudy tools such as *Ansible* and *Terraform*.

Stacking secure Qubes

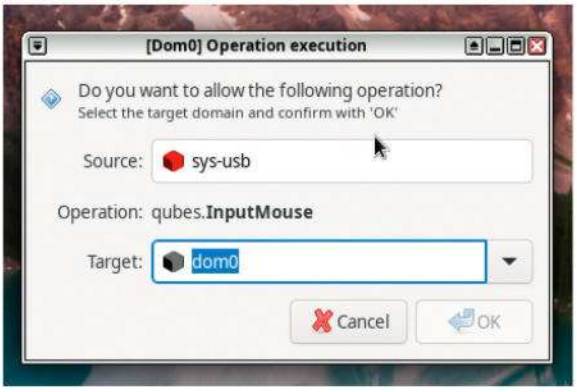
Enough with the waffle, we hear you cry. Fine, let's get this Qubes thing installed and set up.

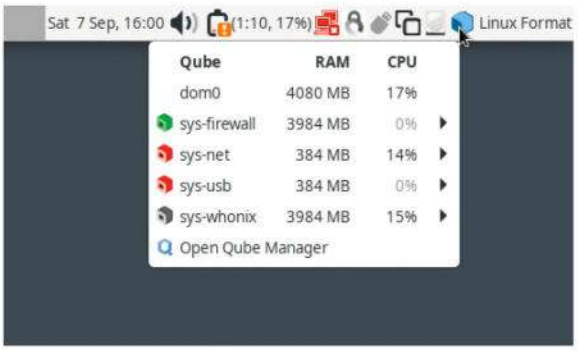
We'll assume you're OK with downloading the Qubes ISO file (from **www.qubes-os.org/downloads**) and writing it to a USB stick. Before you do that, it's worth checking your system meets Qubes's hardware requirements *(see box, as well as the more thorough* **www.qubes-os.org/doc/system-requirements**). We'll also assume you're OK with booting the install medium. There's no live environment to play around in, it just jumps straight into the installer. At the least you have to choose where and how to install Qubes (in the Installation Destination section), as well as create a user account. You might want to disable the Encrypt My Data option, if theft is not part of your threat model (or forgetting your passphrase is). The *Xen* hypervisor is then installed, and here you may break for tea.

All going well, you're invited to reboot the system, then after entering the disk decryption passphrase, you are eventually greeted with... Yet more configurating. Now it is the turn of the individual qubes. By default, a combination of Fedora 40 and Whonix 17 are used for the initial qubes. You might want to change this – for example, switching out a Fedora qube for a Debian one, but we won't. Note how paranoid Qubes is – USB keyboards and mice are not enabled by default. That doesn't mean they won't work, just that you have to give explicit permission if you use them. If you only have a USB keyboard, you may wish to tick the appropriate box here. We won't cover the ins and outs of the LVM arrangement; the default works fine. Once the installation begins in earnest, we'd recommend a very large mug of tea. It takes a while to provision all of these VMs.

Eventually, and hopefully without error, you're greeted with the Qubes login screen. But wait, there's yet more configurating to do. If you installed a Whonix qube (the default), then Qubes asks whether you want to use the Tor network for all of its traffic (the Connect option), just occasionally (Configure) or disable Tor access altogether. Finally you should see a (somewhat busy) Xfce desktop. Check the menu in the top-left if you want to make it even busier. Although first, you might need to get your mouse working.

Qubes doesn't trust USB mice and neither should you. Unless they're yours and you need them for work.



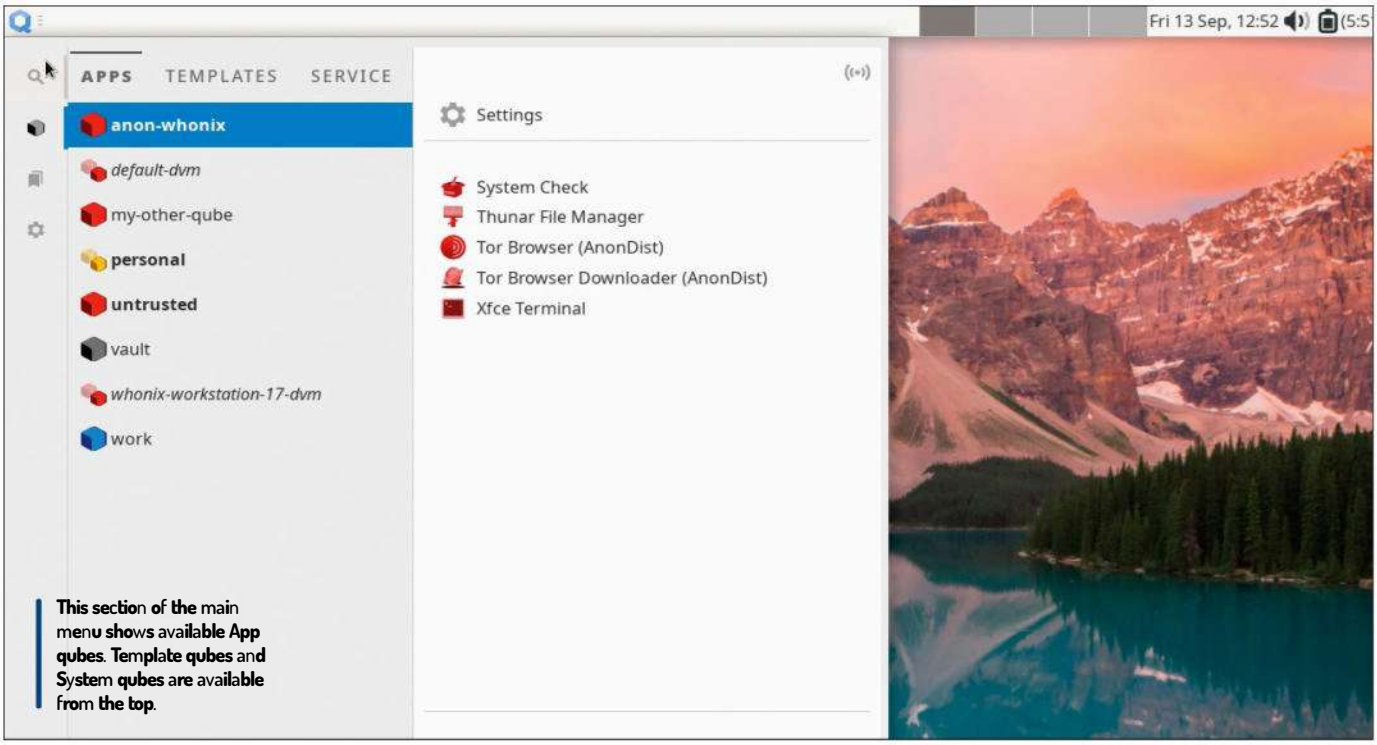


The top-right menu shows running qubes and gives you easy access to the Qube manager.

Qubes worries about so-called Evil Maid attacks, where someone who has physical access to your hardware can tamper with it, in particular by attaching a USB device that may log keystrokes covertly. As such, USB keyboards are disabled by default. Your USB mouse won't work until you accept the prompt to allow the **qubes.InputMouse** operation. Other devices (including storage and our laptop's touchscreen) aren't available until you give them explicit permission from the devices menu (it looks like a USB stick) in the top-right. If you want to use them, attach them to whatever qube you need them in. The default USB behaviour can be changed by going to Qubes Tools > Qubes Global Config > USB Devices.

We've already mentioned the main qubes, but now you can see that some persona-type qubes have also been set up, namely **Whonix**, **Untrusted**, **Personal**, **Vault** and **Work**. These should cover most use cases, and you'll see that each has its own *File Manager, Firefox* and *Terminal*.

There's a bit of subtlety that is quite hard to get your head around at first regarding how things are run in Qubes. The Xfce desktop you see is running on **Dom0**, connected to the (virtual) display manager running on the GuiVM ('VM' and 'qube' are interchangeable in Qubes parlance, but we'll try to be as consistent with what you'd see on the screen as possible). If you then start *Firefox* in the **Personal** qube, that *Firefox* is running on the **Personal** qube, or AppVM, again via the GuiVM. The AppVM inherits its initial filesystem from the Fedora TemplateVM, and system files are immutable (apart from via the **Dom0** system updater – see later). Changes inside each AppVM are stored separately and layered on to the template's filesystem, so they don't interfere with each other and also don't waste space storing redundant data.



If you haven't already, try starting *Firefox* in the **Personal** AppVM. Open the application menu, select the Qubes icon on the left, open the Personal submenu and click Firefox. It'll take a second while the **Personal** qube is started, and then you'll see the *Firefox* you're likely already familiar with. Except this one has a fetching yellow border. If you were to instead use the **Work** qube, it would be blue. The **Vault** (where you might store your darkest secrets) is black. And **Untrusted** qube activities are red – danger, beware.

Click on the blue cube in the top-right (which provides an overview of running qubes) and select Open Qube Manager. This is the central point of administration for all qube-shaped objects. Running qubes have a green circle next to them. You'll see **Dom0** at the top with an important-looking crown on its icon. As mentioned, the app qubes are built off the template VMs. They use the template's root filesystem and store only user data. Conversely, the templates keep the root filesystem persistent (and updated) and store no user data. This is reflected in the disk usage column – the template VMs are gigabytes and the persona ones are much smaller (zero before you start them).

Setting up Tor

You might have told Qubes to use Tor as the default network for the **sys-whonix** qube. And now you might want to force other qubes to use Tor. You can change each qube's networking by changing the NetVM (yes, network access all takes place in another qube, possibly more) in the Qube Manager. Tor access is provided by the **sys-whonix** VM, whereas regular network access is done through **sys-firewall**. A good setup (and the default) is to have only the **anon-whonix** qube use **sys-whonix** networking. Just right-click a qube and select Network to change how it connects. If you'd like to see the Whonix qube's Tor configuration screen again, open the application menu and under the Service heading (in the qubes section), choose Sys-Whonix > Anon Connection Wizard.

And just before this page ends, a cautionary tale. If you were brave enough to install Qubes alongside other OSes, be careful of an odd issue (which befell us) when booting from another distro's boot menu. This results in the **Dom0** kernel being loaded without the *Xen* hypervisor to guide it. And you end up with an empty desktop and, if you dig deep enough, errors about the **qubesd** service not being able to start. It is a very perplexing situation in which to find oneself. The solution is to ensure your UEFI is configured to boot the Qubes image, rather than another distro.

» HARDWARE REQUIREMENTS

Running several VMs at once is an onerous task, so you'll want as much RAM as possible. The official minimum is 6GB but you can easily run this out by starting a few VMs and running a browser in each one. At least 32GB of disk space is required but, again, more is better. Even a simple Qubes setup might end up with four desktop Linux VMs. Imagine if each of those grew to the size of your current install. CPU-wise, anything with a few cores from the last decade will work. Well sort of.

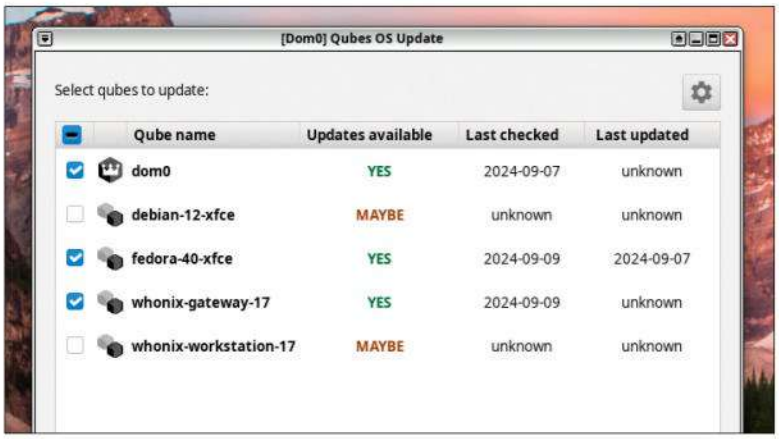
And here comes the tricky part. In order to do all this virtualisation magic at pace, Qubes requires your CPU to have some special virtualisation extensions. And that these be turned on in the UEFI. These are known on Intel as Vt-d and on AMD as AMD-Vi, and sometimes on both as IOMMU. These are in addition to the more common Intel Vt-x and AMD-V. Once activated, you can check they are working as expected (in any Linux flavour) by looking for DMAR (DMA Remapping) entries in the kernel log with `dmesg | grep DMAR`.

If they don't show up there, you may need to boot with either the **intel_iommu=on** or **amd_iommu=on** kernel parameters. You can run Qubes without these extensions (it warns you if they're not there), but it is much slower and slightly less secure.

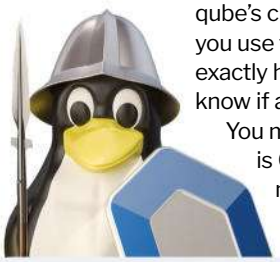
Configure your Qubes

Find out where Qubes differs from other distros and join the single-use society with Disposable qubes.

After fumbling your way around Qubes for a few minutes, you'll likely be impressed. But you'll also likely have noticed that some fairly standard desktop things don't seem to work the same. One of these is copying and pasting, particularly between qubes. The main desktop uses a global clipboard, while the other qubes all have their own private clipboards. So, the usual Ctrl+C and Ctrl+V work as expected while inside a particular qube. But if you want to copy something from one clipboard to another, you need to also use the global clipboard. This is just a matter of some additional digit gymnastics, namely Ctrl+C to first copy to the qube's clipboard, then Ctrl+Shift+C to copy to the global clipboard. Then in the qube you want to paste into, do Ctrl+Shift+V to paste the global clipboard into the qube's clipboard, then finally Ctrl+V to paste. When you use the global clipboard, you'll see a notification of exactly how many bytes are being transferred, so you'll know if any clipboard hijacking malware is afoot.



Just because you have five qubes to update doesn't mean it will be five times as painful as regular updates.



You might want to copy files between qubes – that is OK, but do be careful about, say, taking some malware sample you downloaded in the **Untrusted** qube and copying to somewhere where it could do harm. That would undermine the whole security by isolation thing. Anyway, you can't use regular drag and drop to copy files between qubes, but if you right-click any file in the *Thunar File Manager,* you'll see a Copy To Other Qube option. Which does what you want. Generally speaking, it's better to copy in the direction of higher trust to lower trust, with some obvious exceptions (for example, don't put super-secret files on an **Untrusted** qube).

Also in *Thunar,* you'll find an option to open or edit a file in a Disposable qube. This is prudent if you're for some reason dealing with files that you don't trust. You might want to change things to do with the desktop environment. The background can be changed in the usual right-click the desktop and select Desktop Settings way. All the other options are hidden in the main menu. Select the cog from the left and you'll find the bread and butter options under System Settings.

This cog menu houses all options to do with **Dom0**. If you want to run a terminal, text editor or the like in **Dom0**, the requisite shortcuts are interred in the Other menu. Generally speaking, you shouldn't really be using **Dom0** to store personal things. There's a better qube for that. Sooner or later, you'll need to do some administration on **Dom0**, and there's a plethora of tools in the Qubes Tools menu to do just that.

Isolated updates

You might be notified about available updates in the system tray, or you might be tempted to force the issue by starting *Qubes Update* yourself. Either way, updates in Qubes are a little different. It will surely be a mercy to hear you don't have to update the individual qubes in the traditional way. All you need do (thanks to the clever way the filesystems are built) is update the underlying template VMs. So that means in a default install (where

» DISPOSABLE QUBES

We big-upped Disposable VMs earlier, so let's see how they work. If you look in the **default-dvm** qube menu (it's italicised to indicate its disposablility), you'll see shortcuts for *Firefox* and *Terminal*. Pretty bare-bones, but what do you really need in something throwaway? See over the page if you want more. Open the main menu and find the **whonix-workstation-17-dvm** qube. This is the preconfigured Whonix Disposable qube. There's also a Fedora Disposable (called **default-dvm** and used by the firewall and USB qubes), but this one is more interesting. It contains the *Tor Browser* and *System Check*.

This might tell you that the template needs to be upgraded, in which case you can either follow the terminal instructions or use Qubes's updater. Let's open the *Tor Browser.* This takes a minute while a new qube (called **disp** followed by a random number to indicate its transience) is spun up. The Tor connection itself has already been initiated in the **sys-whonix** qube, so we don't need to wait for that. If you also open a terminal in this Disposable VM, you'll see that another qube (ending in a different number) is spun up. Once you shut both of these down, all trace of them is expunged. These really are single-use tools.

the Debian and Whonix images aren't used), all you need to do is keep the Fedora template updated. This is much more efficient than chugging through the same *Dnf* updates in several terminal windows.

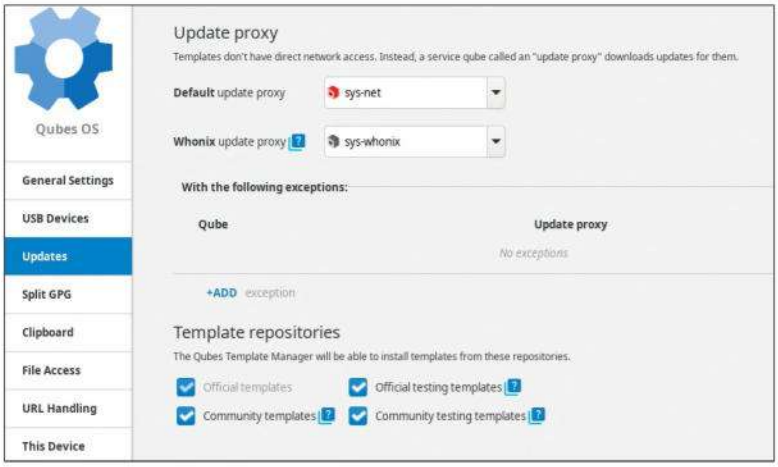
Eagle-eyed readers may have spotted that this process doesn't update **Dom0**. And it turns out this doesn't matter. **Dom0** is isolated from the network VMs, so it's hard to imagine it contracting anything bad. It's also hard to update such an air-gapped machine. If you have a burning desire to do so, though, you should look at the relevant documentation at **www.qubes-os.org/doc/dom0-secure-updates/**. If you're unconvinced, see for yourself by opening a **Dom0** terminal (from that Other menu) and trying to ping your local network. There are always exceptions, though, and occasionally you're prompted to install something from the Qubes Security Bulletin (QSB) via the updater. This might be an update to **Dom0** (for example, the desktop) or the *Xen* hypervisor. Since **Dom0** has no internet, a service proxy is temporarily connected to facilitate data flow.

The included AppVM qubes (**Personal**, **Work** and so on), together with the use of Disposable VMs when you want to leave no trace, should be enough for most people. Otherwise Qubes provides ample mechanisms for making new qubes and customising the bejesus out of them. Or you can clone an existing qube. Or you can use Qubes's powerful template system.

Remember, the template qubes themselves aren't for regular use, they're just there to bootstrap app qubes. Any changes to user directories in an app qube are stored without touching the template, so your personal files and things such as settings and bookmarks persist after a shutdown. So, you can create multiple qubes from the same template without worrying about cross-contamination. And the template itself remains untouched. One possible source of such pollution might be PDF files, which can be malicious. Many a PDF rendering engine has been found vulnerable in the past to carefully crafted documents. Thankfully, Qubes allows you to right-click a PDF file and transform it to a trusted PDF. The result is the original file stripped of any dynamic content, including embedded fonts, vastly reducing any attack surface.

Squaring the software circle

When you modify system directories inside an app qube (for example, by installing some software), those changes are lost when the qube is shut down. The next time it starts, its root filesystem is inherited from the corresponding template. You can use *Dnf* to install anything you need for that session. However, you need to do it again if you need that software again. Naturally, this becomes boring if you want to use the same program a few times.



Additional repositories are configured from Qubes's Global Config tool. You can then run Arch, btw.

If you want applications to persist across reboots, then you either need to install them locally (for example, as a Flatpak or Snap or in your **home** or **/usr/local** directory) or you should modify the template. We mentioned that you're not supposed to use the templates for everyday use, but an exception is made for installing software. Note that this software isn't run on the template qubes themselves, but on the app qubes that are based on them.

One obvious thing that's missing in the app qubes' terminals is tab completion. We can sort that by opening a terminal in the Fedora **Template** qube and running `sudo dnf install bash-completion`. This one hack will make your command-line life (in all the persona qubes) much more pleasant. There are things that you may prefer not to install on the templates, but nonetheless need on multiple qubes (or across multiple sessions). Instead, you can clone a template and install what you need there, and then use that as a template for future, potentially suspect qubes.

The template qubes have almost no network access, but a so-called updates proxy allows *Dnf* (on Fedora) or *Apt* (on Debian) to communicate with their repositories. Note that it's really not recommended to install RPM or DEB files in a template. Not only are you setting yourself up for dependency hell (and having to manually copy those dependencies into the template), but there are also risks associated with being too lazy to check GPG signatures or keep that software updated. If you really must use these kind of packages regularly, it's recommended to install them in a Standalone qube. But anyway, we digress.

You can connect USB devices or internal storage to individual qubes. Microphones might be tricky.

Sending packets over Qubes tubes

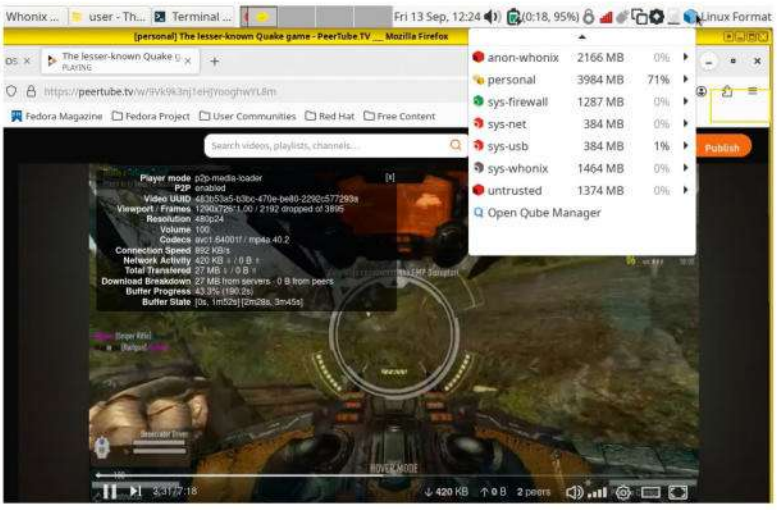
Get Qubes working the way you want, find out how networking works, and start making your own qubes.

Despite each qube's application menu only having a few entries, there are quite a few utilities installed by default. If you want to give them a menu shortcut, go to that qube's settings (in *Qube Manager)* and go to the Applications tab. Most likely you'll want to give *KeePassXC* an icon in the **Vault** qube. Any GUI apps you later install find their way into that Applications tab. You can use the refresh menu button to update the main menu immediately.

Secure networks

Each qube has its own private IP (beginning with 10; you can see them in *Qube Manager)* and the AppVMs can't see or even ping each other (try it). You'll also see in *Qube Manager* that **Dom0** has n/a in its IP address column, as it has no network access. We mentioned the **sys-firewall** (used by everything except Whonix) and **sys-whonix** NetVMs previously. And now if you dig deeper, you can see that the **sys-whonix** qube actually uses **sys-firewall** under the hood. This makes sense as we want Tor traffic to be subject to firewall rules, too. Going one step deeper, you can see that the **firewall** qube uses **sys-net**, which is the qube that actually connects to any Wi-Fi or Ethernet adaptors.

We haven't really talked about what (or who) Whonix is. So let's change that. It's a privacy-focused OS (like Qubes) that runs in a VM. Actually two VMs. A gateway and a workstation. Also like Qubes, the workstation is isolated from the network and relies on the gateway to connect. That gateway (which is what **sys-whonix** is in Qubes) acts as a Tor router. The combination of Qubes and Whonix, then, seems pretty ironclad – as traffic inside a Whonix app qube is about three virtual hops (each in theory adding a layer of filtering/encryption/security) away from any hardware.



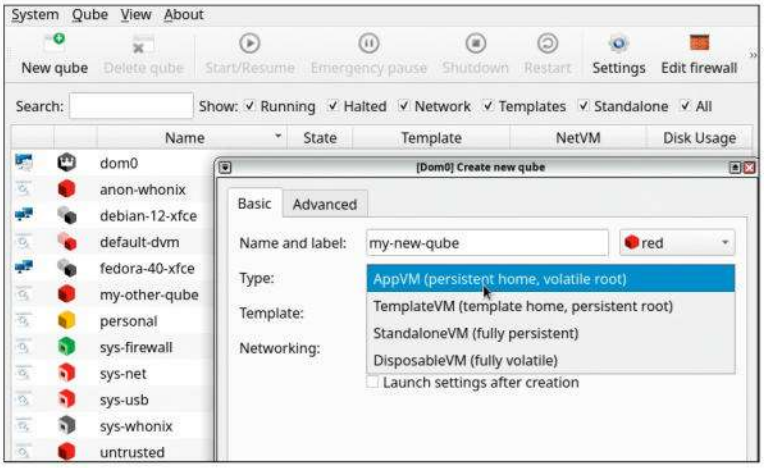
This screenshot cannot convey how choppy video streaming is in Qubes.

The padlock icon in the top-right houses a menu that shows which qubes are using the Tor network. You can run *Sdwdate* (from the Secure Distributed Web Network Time Synchronisation project, see **https://github.com/Kicksecure/sdwdate**) in each of these to ensure time is consistent. Traditional tools like *Ntpdate* don't work well with Tor (and if you're truly paranoid, you wouldn't want to connect to a clearnet NTP server, even over Tor). The small time drifts that happen in your hardware could lead to log files getting confused and time-sensitive encryption keys failing. Before you start the *Tor Browser,* check the accompanying downloader tool in case there's an update.

As a lot of what you might use Qubes for happens in a browser, and as *Firefox* is an excellent choice of browser, we'd like to devote some space to hardening it. Even if you're not using Qubes, you may find this useful. If you haven't already, change the default search engine to DuckDuckGo. You might even want to install *Firefox's* Multi-Account Container extension, to keep different logins to the same site isolated. But perhaps Qubes by its very nature already does this for you.

For maximum privacy (like you're avoiding three-letter agency kind of privacy), you would, of course, use the *Tor Browser.* But there is a community dedicated to customising *Firefox* for maximum privacy so you can still enjoy a conventional browsing experience. At the cutting edge of this effort is the Arkenfox project (**https://github.com/arkenfox/user.js/**), which provides a **user.js** file with every possible paranoia tweak activated. This isn't something you can use out of the box, though (lots of sites don't work with the file).

Something that you can use immediately, though, is *BetterFox* (**https://github.com/yokoffing/BetterFox**), which also provides its own preferences file. Follow the instructions there (especially the backing up your profile part, so that things can be easily restored) to download **user.js** and drop it in your **profile** directory. You'll also want to keep following the instructions to install an ad blocker and possibly enable *NextDNS*, too, to protect against suspect domain names. *BetterFox* also makes several performance improvements and enables smoother scrolling. You can, of course, study

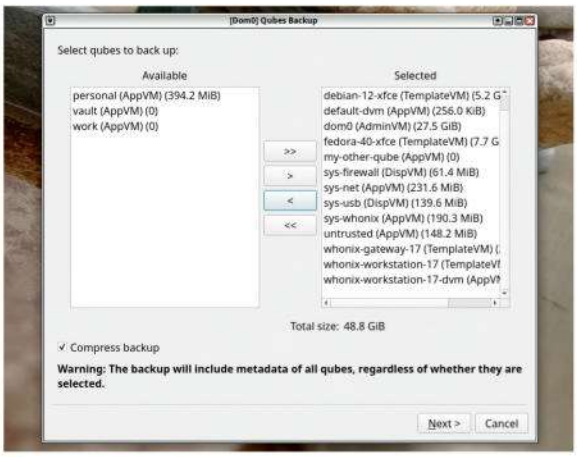


It's amazingly easy to create and configure a new VM. The four kinds of AppVM are succinctly described here.

the documentation and manually disable or further tweak some of these.

Getting *Firefox* configuration changes to stick in a Disposable template is a little nuanced (obviously, getting anything to stick in a Disposable qube based on that template is fruitless). It goes against best practice to run *Firefox* (or anything other than system updates) in a Template qube, since this adds a new profile, which introduces some stateful data. And this could just about conceivably introduce compromise. But Disposable templates are really App qubes, so it's less frowned upon here. There's an interesting thread on this topic at the Qubes forum (**https://bit.ly/LXF321QubesForum**), which we'll leave you to peruse.

Having extolled the virtues of Qubes, we should point out that it's not suitable for all purposes. Qubes never claimed to be a gaming distro, and if it did, it would be a terrible one. There is no video hardware acceleration, it's software all the way down. This also means that watching high-res, full-screen YouTube (or PeerTube) videos is a non-starter. In fact, to watch anything full-screen you need to give the app qube it's running in the appropriate permissions. It is possible to run Qubes with PCIe passthrough to a discrete GPU – see, for example, **https://bit.ly/LXF321QubesPassthrough**. When this works, it enables you to have, for instance, a Windows VM that can run games at very near full speed. A more suitable application for Qubes might be for using the GPU for password cracking or AI modelling. Getting all that working is hardware dependent and far beyond the scope of this introduction.



Backups are easy with Qubes. You needn't worry about backing up the default templates, since these can easily be recovered.

Backup Qubes

It's never too late (as we see the end of the page fast approaching) to back up. And Qubes thoughtfully makes backing up your qubes very easy. You'll find a *Backup Qubes* tool in the Qubes Tools menu. And you can back up to USB devices (once you enable them) or other qubes. You need to supply (and remember) a passphrase, because backups are (of course) encrypted. The *Restore Qubes* tool includes a helpful option to verify the integrity of backup files, so you can ensure they're doing what they're supposed to.

Hopefully, by now you've seen that Qubes is indeed a reasonably secure operating system. But no OS on its own can truly protect you against all threats. And certainly can't protect you from your own (in)actions. Good digital hygiene is pretty much an essential skill for surviving the internet today. This means using strong passphrases, stored in a password manager, applying updates, being incredibly cautious about email attachments (and the princes who send them), and checking the authenticity (using GPG signatures) of any software you download. Qubes doesn't excuse you from any of these duties. If you let your guard down, and the **Dom0** VM somehow gets compromised, it's game over. An attacker would have access to not only your qubes but also (through the hypervisor) your hardware and network. And on that cheery note, we'll leave you to your qubes. LXF

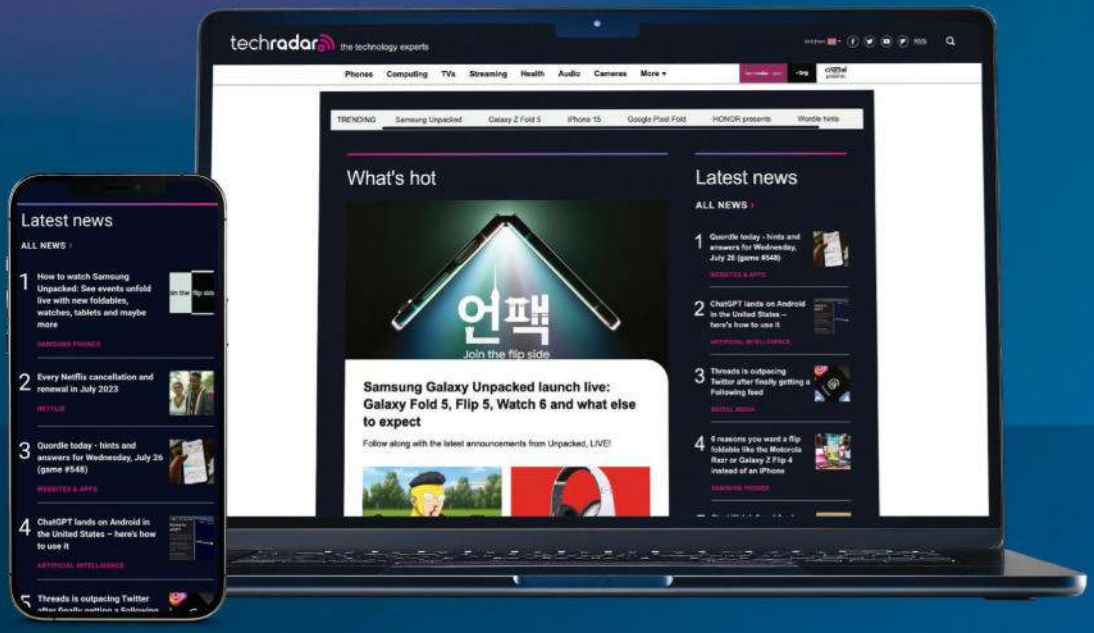
» GLEAMING THE QUBES

If you want to make a new qube, click the button in *Qubes Manager*. We've already seen (and installed) the official templates (Debian, Fedora and Whonix). If you want to see more templates, go to Qubes Tools > Qubes Global Config, select Updates and scroll down to Template Repositories. From there you can enable the Official Testing, Community and Community Testing templates, where you'll find Arch, Gentoo and more templates. The community templates aren't tested by the Qubes team, so care should be taken with them.

There are also some user-contributed templates available. But these can be bulky, so you should also be careful that you don't fill up the root filesystem. By default, this is only 20GB, and ParrotSec (a bulky security-focused distribution with qubes available at **https://qubes.3isec.org/Templates**) will easily fill this up. Fortunately, the default is also to install with LVM (Logical Volume Management), so if you have the space, this is easy to embiggen; see the previous link.

Once you're happy with your template choice, you need to decide whether you want an App qube (with a volatile root and persistent home), a new Template qube (persistent root, volatile home), a Disposable qube, or a Standalone qube (totally persistent filesystem). Standalone qubes are seen as the most risky, because they might be compromised for some time before you notice.

Meet the technology experts



- The world's **most** comprehensive technology website
- An **unrivalled** mix of news, opinions, reviews and features
- **All-new design**, new homepage, new features and special reports
- Backed by **over 300 years** of editorial experience

techradar
the technology experts

www.techradar.com

Free AI courses from Pi Foundation & Google

The hottest current topic can now be picked up by educators for free online.

Rarely does a day go by without AI crossing your path either in the news or via a new product that promises even smarter results thanks to some sort of magic machine learning. In conjunction with Google DeepMind, Pi Foundation educators can now pick up an online course that's all about removing the mystery surrounding AI.

Following the SEAME framework, it breaks down AI into four levels: Social & Ethical, Applications, Models and Engines. Using this, educators can choose which aspects to focus on and the appropriate level, helping to guide classes and discussions.

Run over a three-week period requiring two to three hours per week, this self-paced course covers a range of topics essential for educators. It's part of the Pi Foundation's wider work of bringing coherent, comprehensive learning materials on AI to schools, pupils and educators. Get started now: **https://bit.ly/lxf321learnai**

We'll all have to get up to speed with machine learning and more.

Pi Rust

Embedded stability.

An ongoing project has brought Rust to the latest RP2350, so you can enjoy the most stable of languages on the latest embedded hardware. There's enough to run a basic OS but more support work is needed. Get on board: **https://github.com/rp-rs**

CREDIT: Rust

Now available on the smallest of Pis.

Pico VS Code

Easy coding time.

Towards the end of September, the Foundation released its VS Code Extension to add support for its Pi Pico range. The extension ensures your code is marked up and syntax-correct, and is available in the VS Marketplace as you read this. Get a full guide: **https://bit.ly/lxf321code**

VS Code is widely used and highly supported.

Les Pounder works with groups such as the Raspberry Pi Foundation to help boost people's maker skills.

» THE IMPRESSIVE POWER OF PI

There is a new Raspberry Pi 5 on the block, but this new model is the more wallet-friendly £50 2GB version. On release, the Raspberry Pi 5 came in two versions: the top-tier £80 8GB and the mid-tier £60 4GB. Both of these models are way over the original £35 price tag of the first Raspberry Pi, so the 2GB model offers the power of the Raspberry Pi 5, but at a lower cost.

We've known about the Raspberry Pi 5 2GB (and the future 1GB) since the Pi 5 was announced back in September 2023 – there are silkscreen markings and resistors on the PCB that denote which version that you have.

The Raspberry Pi 5 2GB is just as potent as the 4GB and 8GB. Over at Tom's Hardware (**https://bit.ly/lxf320pi2gb**), I benchmarked the 2GB against the other versions and found that it worked just as well. Sure, it failed the browser tab test with 40 tabs open at once, but it held its own at 10 tabs. So, if you just need to open a few tabs for reference as you hack a project to life, the 2GB is more than enough, rather than trying to use it as a desktop replacement!

High-end emulation is also possible. I tested the Nintendo GameCube emulator, *Dolphin*, with *The Legend of Zelda: Wind Waker* and it ran at a smooth 30fps. *Star Wars Rogue Squadron 2: Rogue Leader* ran awfully, but it isn't much better on my Steam Deck and that has much more power, and costs a heap more.

For AI, machine learning and general high-power projects, the Pi 5 2GB will do the job within its memory limitations.

Recalbox 9.2

Les Pounder travels back to a time when a pocket full of change could open the door to whole new worlds.

IN BRIEF

The latest emulation suite to support the Raspberry Pi 5 brings a world of retro gaming to the comfort of your sofa, with little configuration needed. Despite the Raspberry Pi 5 having the power to emulate 'newer' systems, those emulators are just not ready for *Recalbox*. This is a shame because *Recalbox* looks and feels great.

Emulation has been possible on the Raspberry Pi since it was first released back in 2012. Back then we could emulate up to the Super Nintendo era with relative ease, and some PlayStation 1 games were possible. But with the power of the new Pi 5, we can emulate much more recent consoles, which is where *Recalbox* comes in.

Recalbox 9.2.2 Pulsar is a 64-bit version of the popular retro-gaming console. Installation was easy – we used *Raspberry Pi Imager* to flash a 64GB microSD card. From there we booted into the front-end, *EmulationStation*. *Recalbox* comes configured for keyboard and joypad support. Our Xbox controller was detected via USB and when we triggered the Bluetooth pairing, it was found almost immediately. On reboot, the joypad auto-connected and we could sit back and play *Wipeout 2097*. Until we wanted to exit the game. Usually you would press the Xbox and Start button to exit from the emulator, but it didn't work. We had to get up off the couch and press Escape. This is a config issue and it can be easily rectified via the expansive configuration menu.

Speaking of the config menu, we can scrape box art or screenshots of games from here to make our library look beautiful. We can change the *EmulationStation* theme, but the default vertical scroll is perfect for us. We can also manage sound, network and system updates, all via a joypad. For emulators that require a BIOS (Amiga, PlayStation, Apple and so on), we can use the BIOS checking option to scan the BIOS folders, ensuring we have the files necessary to boot our chosen emulators.

Emulation is easy. Dig out your legal ROMs and once you have them, just insert a blank USB drive into your Raspberry Pi and *Recalbox* formats it ready for use. On your main machine, drop your ROMs into the corresponding system folders, remove the drive, plug it back into the Raspberry Pi running *Recalbox* and it handles cataloguing your library. We did notice that the games ran from the external USB device. We expected *Recalbox* to copy the games on to the microSD card, but this doesn't seem so – rival *RetroPie* can do that.

The Recalbox user interface is made for joypads, with a big and bold aesthetic.

The emulators are where we get the fun from *Recalbox*. With the games in their folders, they appear in the *EmulationStation* menu. If you have the BIOS, you are ready to play. We loaded up *God of War: Chains of Olympus* for the PlayStation Portable (PSP). This game is well known as a stress test in retro gaming. Even the mighty Pi 5 was only able to smoothly run the game at the native PSP resolution (480x272). Doubling the resolution proved too much and frame rates dropped significantly. You can tweak this per emulator, but *Recalbox* has an advanced emulator configuration.

We wanted to test Nintendo Gamecube performance; we've tested the *Dolphin* emulator with Raspberry Pi OS and it works fine. Sadly, there is no Gamecube emulator compatible with *Recalbox* on the Pi 5. The latest generation of consoles we could emulate with *Recalbox* was Sega's Dreamcast. After a quick controller tweak, we were soon driving around in the world of Crazy Taxi.

Recalbox is excellent. It has its quirks, but it is a smooth gaming experience for those of us who want to curate our own gaming museum using a Raspberry Pi 5. We'll wait for the more recent consoles to be supported – that is in the hands of their respective developers. LXF

Les is not a licensed taxi driver, but if he were, then Crazy Taxi taught him well.

VERDICT

DEVELOPER: Recalbox
WEB: www.recalbox.com
LICENCE: Mixed

FEATURES	8/10	EASE OF USE	9/10
PERFORMANCE	8/10	DOCUMENTATION	8/10

A great way to build your first retro-gaming experience with the Raspberry Pi. There are flaws, but they can be fixed.

» Rating 8/10

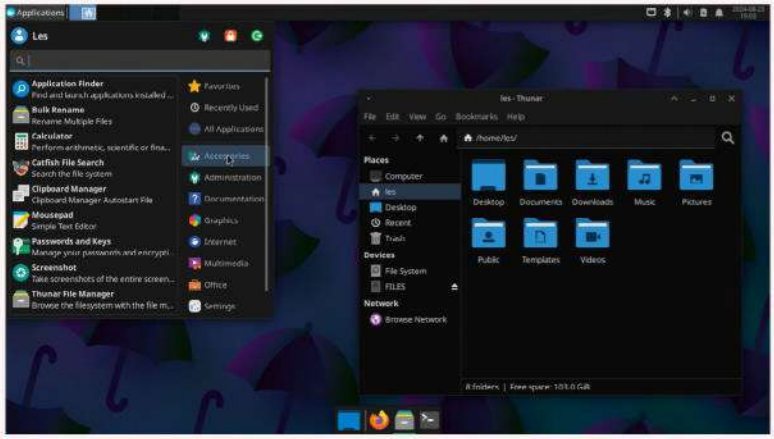
Ultramarine Linux

Les Pounder knows that good looks will only get you so far, and Ultramarine Linux has the looks, but sadly none of the performance.

IN BRIEF

Based upon Fedora, Ultramarine Linux has nothing to do with a *Space Marine* chapter; instead, it's a great-looking, if sadly flawed, Linux distro. Even on an overclocked, USB-to-NVMe-powered Raspberry Pi 4 8GB, it struggled. How it would perform on microSD is something we dared not test.

Ultramarine Linux is based upon the venerable Fedora, and the distro touts itself as an OS for work and play, for all levels of user. It hits the mark in ease of use, but this would not become our daily driver, just yet. You see, Ultramarine Linux is only available for the Pi 3, 4 and 400, all of which have now been superseded by the Pi 5. Ultramarine Linux could really do with the power of the Pi 5, as it runs slow, even on our overclocked (2.1GHz) 8GB Pi 4, running from a USB-to-NVMe drive. The boot time is laboriously slow, taking two minutes and one second to reach the login prompt. Ouch!



Silky blacks and darkness provide something to look at while you wait for anything to happen.

Let's give it a chance, though. It looks great, and the desktop feels great. We chose the Xfce version, another concession for light impact on resources. Booting for the first time, we were asked to create a new user. The UI is clunky, with confirmation buttons in the top-left, and no ability to change keyboard layout during install. Where is the # key again? OK, we can do this once the system boots. The time and date were incorrect, and it seems it wasn't connecting to an NTP server, despite having an Ethernet connection. Some terminal commands later, and we had the correct time, date and time zone.

The Xfce desktop uses a dark theme, and a central app drawer keeps key apps within easy reach. We clicked on *Firefox* to test its web browser performance, then waited for a minute. No feedback, just a dark rectangle where the *Firefox* window should be. We dropped into a terminal (which looks awesome) and killed the *Firefox* process. We reloaded *Firefox* via the terminal, and saw in the output that it couldn't access a section of memory. We abandoned the browser test. Wi-Fi worked as did Bluetooth file transfer, so there is the ability to get online.

Software installation is via *DNF*, and being Debian users, we had to get used to how Fedora worked. It didn't take long, and we installed *Neofetch* to test that we could install packages. Then we did a system update, all 4GB of it via Ethernet. That took some time and at times it locked up. Our internet is fast, so it shouldn't take this long, right? We could see the drive activity light flashing, and the system clock was updating. We were still waiting after 25 minutes. We opened a virtual console (Ctrl+Alt+F1) and logged in. Running *htop,* we could see there was activity, so things were happening, right? No, it crashed and offered no feedback. Rebooting, the system crashed on boot, forcing us to reinstall the OS.

Accessing the GPIO and camera proved to be a non-starter. Via *Bash,* we can usually access the GPIO, but no amount of fiddling helped. The same for the camera. So, we can't recommend this distro for maker projects.

Ultramarine Linux looks great and it is easy to use. But it is also frustratingly slow and clunky, even with a top-spec Raspberry Pi 4 setup. We hit every issue possible during our tests. We wanted to love this distro, we really did. But we didn't have the best user experience with it, sadly. As an alternative to Raspberry Pi OS, Ultramarine Linux misses the mark. A crying shame because it had much promise and potential. LXF

We overclocked our Raspberry Pi 4 8GB to 2.1GHz (Neofetch reported 2.2GHz, but it was wrong) and we still had a slow experience.

```
                    les@ultramarine
                    ---------------
OS: Ultramarine Linux 40 (XFCE Edition) aarch64
Host: Raspberry Pi 4 Model B Rev 1.4
Kernel: 6.8.11-300.fc40.aarch64
Uptime: 6 mins
Packages: 1862 (rpm)
Shell: zsh 5.9
Resolution: 1280x720
DE: Xfce 4.18
WM: Xfwm4
WM Theme: Materia-dark
Theme: Materia-dark [GTK2], Adwaita [GTK3]
Icons: Papirus-Dark [GTK2], Adwaita [GTK3]
Terminal: xfce4-terminal
Terminal Font: Monospace 12
CPU: (4) @ 2.200GHz
Memory: 888MiB / 7775MiB
```

VERDICT

DEVELOPER: Fyra Labs
WEB: https://ultramarine-linux.org
LICENCE: Mixed

FEATURES	5/10	EASE OF USE	5/10
PERFORMANCE	4/10	DOCUMENTATION	7/10

A gorgeous-to-look-at but deeply painful Linux experience – even on top-spec hardware, it crawls along.

» Rating 5/10

HEAT SENSORS

Work with Pi Pico temperature sensors

We always knew **Les Pounder** was hot stuff and he now he can prove it with a myriad of temperature sensors attached to his spare Pico.

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his adventures and projects at **http://bigl.es**.

Back in **LXF318** we took a look at sensors that could detect movement and distance. This month, we are looking at sensors that can measure temperature and humidity (in the case of the DHT11). We'll be using MicroPython on a Raspberry Pi Pico, but you can also use a Raspberry Pi Pico W.

The goal is to learn how these sensors work and how we can integrate them into our Raspberry Pi Pico and Pico W builds. We'll be using *Thonny* to write the project code. An assumption is made that you know how to use *Thonny* with the Raspberry Pi Pico; if not, Tom's Hardware has a guide: **www.tomshardware.com/how-to/raspberry-pi-pico-setup/**.

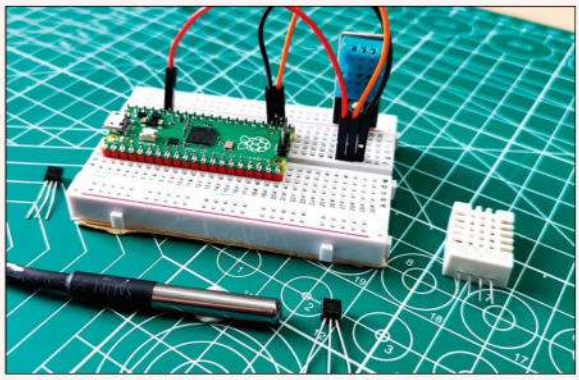
The DS18B20

Can you have a favourite sensor? If so, this is ours. The DS18B20 is a classic temperature sensor that comes in a variety of forms. Made by Maxim, but there are many clones, the DS18B20 is a versatile temperature sensor that we have personally used in many projects. Its versatility is demonstrated by the various packages it comes in. You see, the DS18B20 is a three-pin chip, but it can be contained inside a waterproof container and submerged in water. If you are going to buy any DS18B20, buy one of these waterproof versions. Want to see the temperature decay rate for a cup of tea? Drop the DS18B20 into a fresh cuppa, record the data to a CSV file, drop it into a spreadsheet, et voila!

Let's get to grips with the DS18B20, and we'll start with the circuit. The DS18B20 has three pins. Looking at the flat face of the chip, we have this pinout:

DS18B20 pin	Wire colour	Function	Pico pin
Left	Black	GND	Any GND
Centre	Yellow	Data Out	GPIO17
Right	Red	VDD (3-5V)	3V3 Out

The circuit is simple, but there is one tricky bit. The VDD pin of the DS18B20 is connected to 3V3 Out on the Raspberry Pi Pico, and the GND pin of the DS18B20 is connected to any GND on the Pico. The centre data pin connects to GPIO17 on the Pico, but you will notice a resistor between this pin and the VDD. This is a 4.7K resistor, which pulls the data pin high



Here are all of the temperature sensors that we used in this tutorial – with very little code, we can get reliable data.

using the 3.3V output from the Pico. The data pin must be kept high in order for the Pico to read the data. Please refer to the circuit diagram in the download for this project, follow the connections and you'll have a working circuit in a few minutes.

Luckily for us, MicroPython has a module for the DS18B20 baked into its firmware. Open *Thonny* and connect your Raspberry Pi Pico. In the blank document we'll start writing the code, which begins with importing modules for GPIO access, onewire serial communications, the DS18B20, and timing control:

```
import machine
import onewire
import ds18x20
import time
```

Now, let's tell the Raspberry Pi Pico where the DS18B20 data pin is, in this case GPIO17. Then we'll create an object to use the DS18B20 sensor, passing the GPIO pin reference via the onewire serial module. This tells the Raspberry Pi Pico where our DS18B20 is, and how it should talk to it:

```
ds_pin = machine.Pin(17)
ds18b20_sensor = ds18x20.DS18X20(onewire.
OneWire(ds_pin))
```

Creating an object called **roms**, we get the address of the DS18B20 on the onewire serial bus. We'll need this to get the temperature data. We can have multiple DS18B20s on the same onewire bus, but they all need different addresses. Many clones have duplicate

addresses, causing conflicts when many are used on the same bus.

```
roms = ds18b20_sensor.scan()
```

Inside a **while True** loop, we call **convert_temp**; this must be done before we can read the temperature:

```
while True:
  ds18b20_sensor.convert_temp()
```

A short one-second pause occurs before we start a **for** loop. Inside the **roms** object there could be multiple DS18B20s, in our case there isn't, but the **for** loop iterates over every DS18B20 sensor it can see.

```
  time.sleep(1)
  for rom in roms:
```

Now we can read the temperature, stored by default in degrees Celsius. Store it in an object called **tempC**:

```
    tempC = ds18b20_sensor.read_temp(rom)
```

Now let's show the temperature to the user. We'll print it to the Python shell (REPL) using string formatting to insert the temperature in a sentence. The **:..2f** rounds the value down to two decimal places. Finally, outside the **for** loop but inside the main loop we add a five second-pause before the process repeats:

```
    print('The temperature is:', "{:.2f}°C".format(tempC))
  time.sleep(5)
```

Save the code to the Pi Pico as **ds18b20-test.py** and click the play button to start the code. You should see the temperature data appear in the Python shell.

The TMP36

The TMP36 is an analogue sensor that uses voltages to represent the temperature. This means we have to do more work to get the correct temperature data, but it is worth it. The TMP36 comes in a three-pin chip package. Looking at its flat front, the pins are.

TMP36 pin	Wire colour	Function	Pico pin
Left	Red	VDD (2.7-5.5V)	3V3 Out
Centre	Yellow	Data Out	GPIO26
Right	Black	GND	Any GND

The circuit is much simpler than the DS18B20. There are only three connections made between the Pico and TMP36. See the circuit diagram in the download for more details.

As this is an analogue sensor, we'll convert the returned values into something that we understand. So, let's get started in a new blank *Thonny* window. We'll import the **ADC** class from machine used to convert analogue to digital (A,D,Convert) and the time module to control the pause in our loop:

```
from machine import ADC
import time
```

Setting up the **adc_pin** and **tmp36** basically tells the Pico that we have something connected to GPIO26, and that it is an analogue component. The **tmp36** object makes it easier for us to understand what it is.

```
adc_pin = 26
tmp36 = ADC(adc_pin)
```

Inside of a **while True** loop, we start the temperature sensing part of the code by reading the raw ADC value of the TMP36 on GPIO26. The TMP36 output voltage will change and the ADC will read this as a value between 0 and 65535.

```
while True:
  adc_value = tmp36.read_u16()
```

Now let's do some maths. The value of **v** (volts) is going to be the product of 3.3 (the 3.3V voltage we are feeding the TMP36), divided by 65535 (the 12-bit resolution of the Pico's ADC). Then we multiply the output of this by the TMP36's returned **adc_value**:

```
  v = (3.3/65535)*adc_value
```

To convert this to degrees Celsius, we need to do a little more maths. The **tempC** object will store the output of 100 multiplied by the value stored in our object, **v**. Then we take 50 from that value.

```
  tempC = (100*v)-50
```

Using the same string formatting sentence structure as we used in the DS18B20 code, we print the temperature to two decimal places. Then the code pauses for five seconds before the process repeats:

```
  print('The temperature is:', "{:.2f}°C".format(tempC))
  time.sleep(5)
```

Save the code to the Raspberry Pi Pico as **tmp36-test.py** and click on the green play button to start the code. You should see the temperature data appear in the Python shell.

YOU NEED

- **Pi Pico or Pico W**
- **DS18B20**
- **TMP36**
- **DHT11**
- **4.7K ohm resistor (yellow-purple-red-gold)**
- **3x M2M jumper wires**
- **Breadboard**
- **Get the code: https://bit.ly/lxf321**

THE DHT11 AND DHT22

OK, one more sensor. The venerable DHT11, the plastic-blue-cage-looking sensor (or DHT22 with a white cage), can do temperature and humidity. The DHT11 has four pins, but only three are needed. Refer to the circuit diagram for the connections.

In *Thonny,* the code is simple. In a new file, import the **Pin** class to use the GPIO, then **time** for controlling the loop. Finally import **dht** to enable use of the sensor:

```
from machine import Pin
import time
import dht
```

Create a **sensor** object to store the GPIO connection to the DHT11:

```
sensor = dht.DHT11(Pin(17))
```

In a loop, we pause for two seconds and then take a temperature reading:

```
while True:
  time.sleep(2)
  sensor.measure()
```

Save the current temperature and humidity to their respective objects:

```
  temp = sensor.temperature()
  humidity = sensor.humidity()
```

Finally, using the same string formatting that we have used previously, print the temperature and humidity in a sentence. The humidity is rounded to one decimal place:

```
  print('The temperature is:', "{:.2f}°C".
format(temp))
  print('The humidity is:', "{:.1f}%".
format(humidity))
```

Save the code as **dht11-test.py** and click on the play button to start the code. You should see the temperature data appear in the Python shell.

RGB LEDS

Power projects with the best Pi Pico LEDs

We always knew **Les Pounder** was a bright spark and he proves it with the brightest possible LEDs!

OUR EXPERT

Les Pounder is associate editor at Tom's Hardware and a freelance maker for hire. He blogs about his adventures and projects at **http://bigl.es**.

LEDs are the gateway to electronics. From the humble single-colour LED to the exotic RGB LEDs, we've tried them all and love them all equally. But what if you want to start using WS2812B or APA102? Also, what do those mixtures of letters and numbers mean?

The goal of this tutorial is to learn how these RGB LEDs work and how we can integrate them into our Pi Pico-powered builds. We'll be using *Thonny* to write the project code. An assumption is made that you know how to use *Thonny* with the Raspberry Pi Pico. If not, Tom's Hardware has a guide at **www.tomshardware.com/how-to/raspberry-pi-pico-setup/**.

A NeoPixel world

WS2812Bs, more commonly known as NeoPixels, are addressable RGB LEDs that have been seen in a myriad of Pi projects over the years. Why? Because they are easy to work with and can be found rather cheaply via the usual online sellers. NeoPixels are addressable, and that means we can control an individual pixel (the common name for one LED in the chain) in a long series of pixels. The brightness and colour of a pixel can be set by the user. With some clever coding, you can make simple animated sequences. We're going to wire up a length of NeoPixels to a Raspberry Pi Pico 2, and write some code to make random groups of LEDs light up in a random colour. Luckily for us, MicroPython now includes the NeoPixel module, so there is no software installation necessary.

Before we can do any software, we need to wire up the NeoPixels. We're using a NeoPixel ring with 12 LEDs, but you can use any type of NeoPixels. The connections are as follows:

NeoPixel pin	Wire colour	Function	Pico pin
IN (Data In or DI)	Yellow	Data connection	GPIO16
PWR	Red	3.3V	3V3 Out
Right	Black	GND	Any GND

Open *Thonny* and connect your Raspberry Pi Pico. In the blank document, we'll start writing the code, which begins with importing modules for GPIO access (machine), NeoPixels, speed control (time) and pseudo-random numbers (random):

```
import machine, neopixel, time, random
```

Next, create an object called **np**. This object is how we interact with the NeoPixels, so we need to tell the Pico to which pin the NeoPixels data connection is made, and how many LEDs there are. In this case, it is GPIO16 and there are 12 NeoPixels:

```
np = neopixel.NeoPixel(machine.Pin(16), 12)
```

A **delay** object is used to store a float value of 0.2. This refers to a 0.2-second delay, which shall be used to control the speed at which the pixels update:

```
delay = 0.2
```

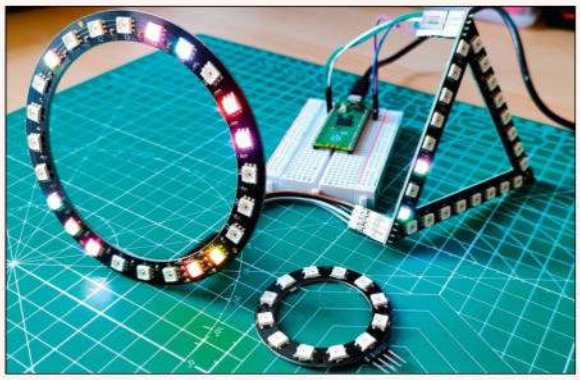
Inside of a **while True** loop, a loop that runs for ever, use a **for** loop to update three randomly chosen pixels:

```
while True:
  for i in range(3):
```

To choose a random pixel, use the **randint()** function from the random module. We have to specify a range, starting from zero and ending one less than the number of pixels on your NeoPixels. Why one less? Because we start counting from zero:

```
    rand_pixel = random.randint(0,11)
```

Create an object to store a randomly generated colour made up of randomly chosen values between 0 and 254. The three randomly generated values are stored in a tuple data object, and refer to the mix of red, green and blue light that each pixel can generate. Zero is off and 254 is full brightness. Actually, 255 is



NeoPixels, DotStar and RGB LEDs are all different types of multicolour LEDs that we can control using just a few lines of MicroPython.

full brightness, but we need to go one value lower due to starting the count at zero:

```
    rand_colour = (random.randint(0,254),random.randint(0,254),random.randint(0,254))
```

Add a short pause to the code:

```
    time.sleep(delay)
```

Using the **np** object that we created earlier, set the randomly chosen pixel to the randomly chosen colour. This isn't truly random, but enough for this project. We then write to the pixel, updating its colour:

```
    np[rand_pixel] = rand_colour
    np.write()
```

Outside of the **for** loop, add a short delay to stop the colour change from happening too quickly:

```
  time.sleep(delay)
```

Using another **for** loop, we turn off every pixel in the NeoPixel chain. We do this by setting each colour to 0. A pause at the end stops the animation happening too quickly before the main **while True** loop repeats:

```
  for i in range(12):
    np[i] = (0,0,0)
    np.write()
  time.sleep(delay)
```

Save the code to the Pi Pico as **np-test.py** and click Run > Run Current Script, or press the green play button to start the code. You should see the NeoPixels light up, three at a time, before they turn off. The sequence repeats until we turn it off.

You're our DotStar

Similar to NeoPixels, APA102 DotStars are addressable RGB LEDs that use two-wire communication via SPI. The connection is faster than NeoPixels, making them useful for projects that use persistence of vision. We're going to use some APA102 RGB LEDs in a similar way to the previous NeoPixel project.

First we need to wire them up. We're using a series of DotStar shapes, but this code will work with any APA102 DotStars.

DotStar pin	Wire colour	Function	Pico pin
GND	Black	GND	Any GND
IN (Data In or DI)	Yellow	SPI SCK	GPIO2
CI	Green	SPI MOSI	GPIO 04
PWR	Red	3.3V	3V3 Out

Open *Thonny* and connect your Raspberry Pi Pico. Before we write any code, we need to copy the MicroPython module code from the amazing MicroPython ambassador Matt Trentini (**https://bit.ly/LXF321MicroPython**) and paste it into a blank *Thonny* project. Save the code to the Raspberry Pi Pico as **micropython_dotstar.py**. Now create a new blank project.

In the blank document, we'll start writing the code, which begins with importing modules for speed control (time), pseudo-random numbers (random) and DotStar, and the **Pin** and **SPI** classes from the machine module.

```
import time, random, micropython_dotstar as dotstar
from machine import Pin, SPI
```

Create an object called **spi** and use it to store the connection point between the Pico and the DotStar LEDs. We need to specify the SPI channel ID (0) and tell the Pico which SPI pins we are using. Note that miso is not used, but we need to tell the code to use it:

```
spi = SPI(0, baudrate=100000, polarity=0, phase=0, sck=Pin(2), mosi=Pin(3), miso=Pin(4))
```

Next, create an object to directly control the DotStar, passing the SPI details, the number of pixels (48 for us) and the brightness. Keep the brightness low for now. Then create an object to store the number of DotStars in the chain – this gets the length from the **dots** object. Then create a **delay** object to add a pause to the code.

```
dots = dotstar.DotStar(spi, 48, brightness=0.2)
n_dots = len(dots)
delay = 0.1
```

Inside a **while True** loop, we use a **for** loop that iterates 12 times (this is a quarter of the LEDs in our chain; change this to match your setup):

```
while True:
  for i in range(12):
```

In the **for** loop, we pick a random pixel, remembering to subtract one from the total number for the range. We then create a random colour. Then we use the **dots** object, passing the random pixel number and colour to alter that DotStar pixel:

```
    rand_pixel = random.randint(0,n_dots-1)
    rand_colour = (random.randint(0,254),random.randint(0,254),random.randint(0,254))
    dots[rand_pixel] = rand_colour
```

We then add a pause, before creating another **for** loop that will iterate over every DotStar LED, turning each LED off before another delay. The main **while True** loop then repeats the process:

```
  time.sleep(delay)
  for i in range(n_dots):
    dots[i] = (0,0,0)
  time.sleep(delay)
```

Save the code to the Raspberry Pi Pico as **apa102-test.py** and click Run > Run Current Script, or press the green play button to start the code. You should see the DotStars light up, 12 at a time (for us), before they turn off. The sequence repeats until we turn it off.

YOU NEED

- **Raspberry Pi Pico 2, Pico,or Pico W**
- **Half breadboard**
- **4x male-to-male jumper wires**
- **Length of WS2182B NeoPixels**
- **Length of APA102 Dotstar**
- **3x 100-ohm resistors (brown-black-brown-gold)**
- **1x RGB LED (common cathode)**
- **Get the code: https://bit.ly/lxf321led**

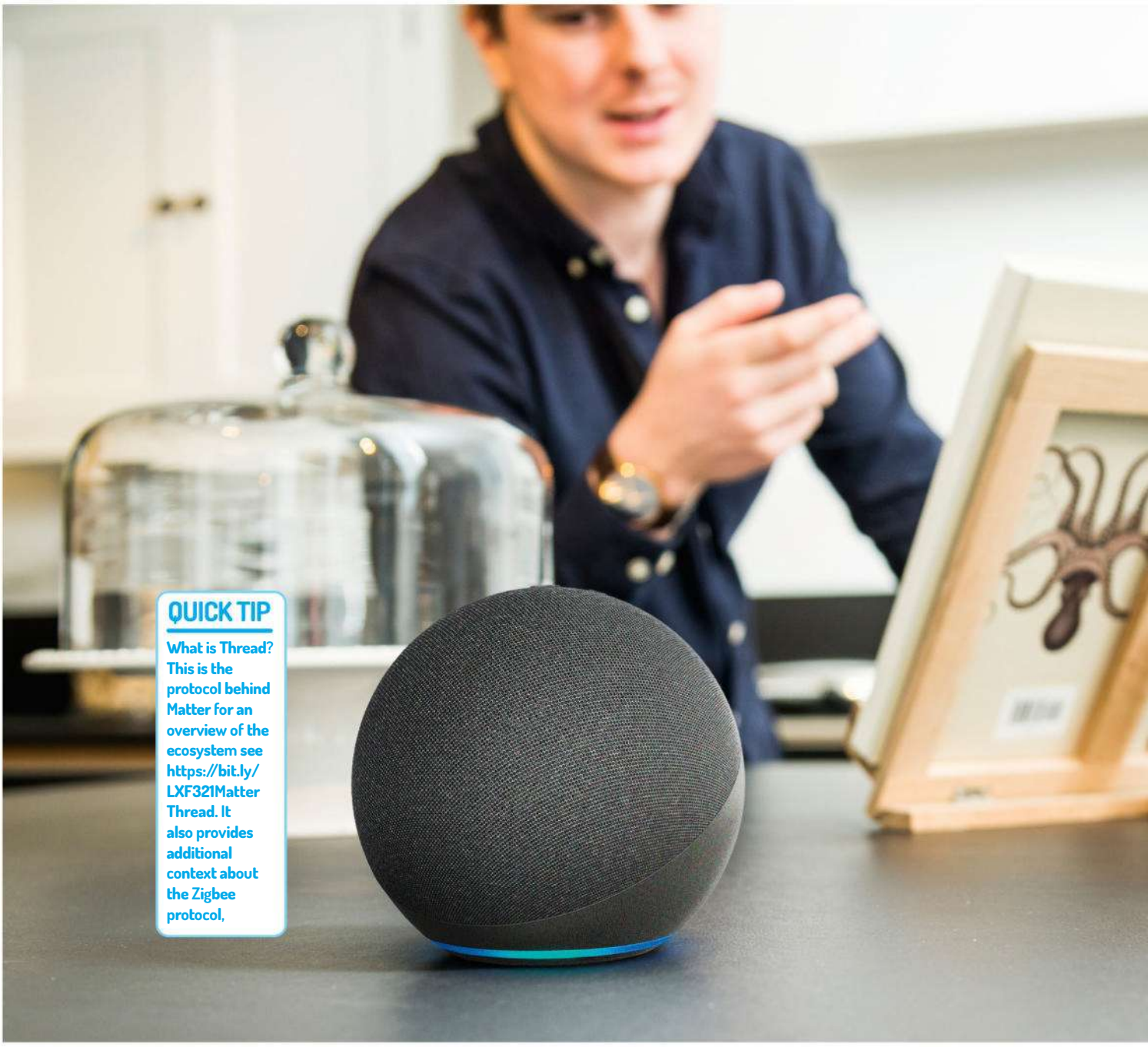
» ANY MORE RGB LEDS?

There are many more forms of LEDs we can play with. A common (pun intended) inclusion in many electronics kits are RGB LEDs. These LEDs come in common anode or cathode configurations. A common anode shares a positive voltage to all three R, G and B LEDs inside the main LED. A common cathode shares a GND connection to all the RGB LEDs. By controlling the state of the RGB pins, pulling them low for common anode and high for cathode, we can turn the individual elements of the RGB LEDs on and off. So, one LED and three colours. No! Using some fancy pulse width modulation (PWM), we can make these cheap RGB LEDs any colour we wish (sort of).

In the download for this tutorial, we have added two extra projects. First, **rgb_led.py** shows how to swap between red, green and blue elements of the RGB LED. For colour mixing, we have included **rgb_pwm.py**, which uses PWM to mix the RGB elements into new colours. Use a comment in the code to turn off a colour – for example, to turn off the green LED do this in both **for** loops:

```
#g_pwm.duty_u16(duty)
```

Run the code and see what colours are created. These RGB LEDs are cheap but provide a challenge from typical single-colour LEDs.



QUICK TIP

What is Thread? This is the protocol behind Matter for an overview of the ecosystem see https://bit.ly/LXF321Matter Thread. It also provides additional context about the Zigbee protocol,

While smart-home projects initially fell under the domain of a wide variety of proprietary protocols, standardisation pressure has led to the emergence of a few dominant interfaces.

The combination of the logical Matter protocol and its accompanying Thread transport layer recently became available in the form of the affordably priced Arduino Nano Matter Community Preview.

Makers and home tinkerers can create Matter/Thread-based smart-home peripherals without having to perform cost-intensive PCB design using a microcontroller or wireless module provided by one of the established vendors in the Thread/Matter space.

The much-liked *Arduino IDE* and its hardware abstraction layer, commonly known as Arduino Core, have established themselves in the IoT space. The availability of the Matter library enables developers to integrate their Arduino-derived knowledge with a Thread-based network interface.

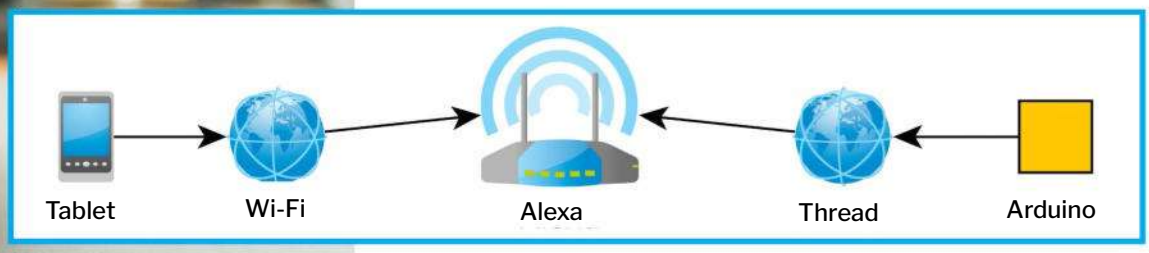
As is the case for most complex communication protocols, much of the effort lies in the initial 'bring-up' of the system. This article will use an Arduino Nano Matter Community Preview PCB (for details see https://labs.arduino.cc/en/labs/nano-matter) and an Amazon Alexa to create a Matter/Thread-based system in the Alexa ecosystem. The steps can also be applied to other vendors' IoT ecosystems with minimal modifications. Be aware due to the advanced nature of this subject, we're presuming prior knowledge of using the Ardunio ecosystem and tools.

Thread bare essentials

Even though Thread uses IPv6 for node addressing, the underlying wireless system is not entirely communal with Wi-Fi. Due to that, a Thread border

OPEN PROTOCOLS MATTER

Google and Amazon might be behind the latest IoT protocol, but Tam Hanna is just the guy to bring it to the masses, with a little help from Arduino...



The Alexa in the system acts as a Thread border router.

router device is required to act as a bridge in a fashion similar to the one shown *(above)*.

If purchasing an Alexa (or any smart hub) specifically for this article, take care – only the spherical Echo 4th Generation has Thread for the router capability. Most other models are not suitable for this project. Be that as it may, pair your Alexa to a Kindle or other Android device *(see Quick Tip, p.51)* before proceeding.

For the following steps, use a laptop on the same Wi-Fi network as the Thread border router is on. In terms of the Arduino development environment, using both *Arduino IDE 2* and the *Arduino Cloud Editor* is possible – the following steps will take place on the workstation using Ubuntu 22.04LTS, where version 2.3.2 of the *Arduino IDE* has been deployed.

The *Arduino IDE 2.0* continues to use the board-module manager initially contributed by Microchip. Because of that, you need to begin by clicking File > Preferences to open the integrated development environment settings.

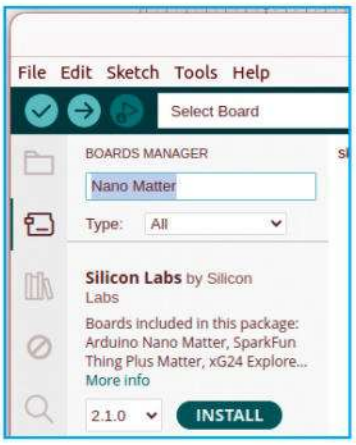
Next, click the text box next to Additional Boards Manager URLs and enter the URL **https://siliconlabs.github.io/arduino/package_arduinosilabs_index.json**. This way, the board package manager is informed about an additional repository containing packages useful for Arduino development. After that, click OK to close the preferences window.

Next, open the board manager and enter the string `Nano Matter` to reveal the Silicon Labs core *(see image, over page, top-left)*. Click Install to deploy it – at the time of writing, version 2.1.0 was current.

After the Arduino core has been downloaded, a Hello World example can be run – this, however, is mainly intended to verify the behaviour of the toolchain. Developers upgrading from older Arduinos should furthermore be careful, because the plug on the Arduino Nano Matter is a USB-C plug.

Analysing Matter

As with many other IoT systems, seeing concepts in an en vivant state is the easiest way to learn more. In the case of the Arduino Core for the Silicon Labs platform, Silicon Labs provides about a dozen examples' worth



Given that the Arduino Nano Matter Community Preview is based on the Silicon Labs chip, the generic Silicon Labs core contains compilers and other relevant tooling.

of code. Sadly, there are known (see **https://bit.ly/lxf321bug**) bugs in the *Arduino IDE* that make accessing them somewhat difficult.

Due to this, first, click File > New Sketch to order the opening of a fresh *Arduino IDE* window. Next, use the Boards option to select the Arduino Nano Matter Community preview – as of this writing, the correct path is Silicon Labs > Arduino Nano Matter. Finally, the Examples menu should populate with options *(see screenshot below)*.

We will use the example **nano_matter_lightbulb_color**, as it has achieved (partial) CSA (Connectivity Standards Alliance, the overseeing board) certification. Select it in the Example menu to open the code in the code editor. After the project skeleton has been rehydrated, look at the header structure:

```
#include <Matter.h>
#include <MatterLightbulb.h>
...
MatterColorLightbulb matter_color_bulb;
```

MatterColorLightbulb is one of about a dozen object representations. The Matter standard, as implemented in the Arduino library, doesn't permit arbitrary smart-home devices; instead, device type is limited to these options (**https://bit.ly/lxf321types**):

- Air quality sensor
- Contact sensor
- Door lock
- Fan
- Flow measurement
- Humidity measurement
- Illuminance measurement
- On/off light bulb
- Dimmable light bulb
- Colour light bulb
- Occupancy sensor
- On/off plug-in unit/outlet

QUICK TIP

Save money! Development boards are high-margin products, where shopping around can save a bit of money. In the case of the Arduino Nano Matter, www.oemsecrets.com/compare/ABX00112 provides a good start.

The Silicon Labs core ships with an ample complement of samples.

- Pressure measurement
- Switch
- Temperature measurement
- Thermostat
- Window covering

Our example implements an RGB light bulb that can be simulated via the RGB LED on the PCB. The initialisation of the sketch then starts out as we would expect it to:

```
void setup()
{
  Serial.begin(115200);
  Matter.begin();
  matter_color_bulb.begin();
```

Invoking the various **begin** methods is not the end of the initialisation process. Instead, the Matter/Thread protocol combination requires the checking of the commissioning state of our peripheral:

```
if (!Matter.isDeviceCommissioned()) {
  ...
  Serial.printf("Manual pairing code: %s\n", Matter.
getManualPairingCode().c_str());
  Serial.printf("QR code URL: %s\n", Matter.
getOnboardingQRCodeUrl().c_str());
}
while (!Matter.isDeviceCommissioned()) {
  delay(200);
}
Serial.println("Waiting for Thread network...");
while (!Matter.isDeviceThreadConnected()) {
  delay(200);
}
```

In a fashion not dissimilar to Bluetooth pairing, devices need to be associated with their counterparts. This happens via a manual pairing code or a QR code, both generated by the Arduino Matter library. Our code example emits the credentials into the Arduino serial terminal.

After that, we find the usual waiting required to ensure commissioning and network connectivity. The rest of the **init** method – we will not print it here due to space constraints – is responsible for housekeeping tasks.

The next exciting part resides in the **loop** function. It starts by analysing the state of the Arduino and performs a few modifications against the actual device state held in the **MatterColorLightbulb** class introduced above:

```
void loop()
{
  if (button_pressed) {
    button_pressed = false;
    matter_color_bulb.toggle();
  }
  static bool matter_lightbulb_last_state = false;
  bool matter_lightbulb_current_state = matter_color_
bulb.get_onoff();
```

Our example is simple, because it uses the integrated RGB LED on the Arduino board. In practical cases, colour emission is usually more complex – think of external hardware such as an LED1202 IC or a WS2812B chain. Due to that, the **MatterColorLightbulb** class limits itself to acting as setting storage – writing out the actual parameters is the task of the developer alone.

In the case of our sketch, we start by invoking methods such as **matter_color_bulb.get_hue()** to determine the values currently stored. These are then compared against stored values, with an update of the hardware state performed if it is so required:

```
static uint8_t hue_prev = 0;
...
uint8_t hue_curr = matter_color_bulb.get_hue();
...
if (hue_prev != hue_curr || saturation_prev !=
saturation_curr || brightness_prev != brightness_curr) {
  update_led_color();
  hue_prev = hue_curr;
  ...
 }
}
```

At this point, the example is ready to run. Connect the Arduino to the workstation using a USB cable, and ensure that the *Arduino IDE* is able to detect the newly connected peripheral device In some cases, program deployment fails with an error along the following lines:

```
efm32s2_dci_read_se_status
Error: unable to open CMSIS-DAP device 0x2341:0x72
Error: unable to find a matching CMSIS-DAP device
```

Should this happen to you, the first remedial step involves using the Sketch > Upload menu. Should this not work, perform the following three commands to update the UDEV rules. This is often required, because updating the UDEV rules doing pack installation in the *Arduino IDE* is known to be flaky:

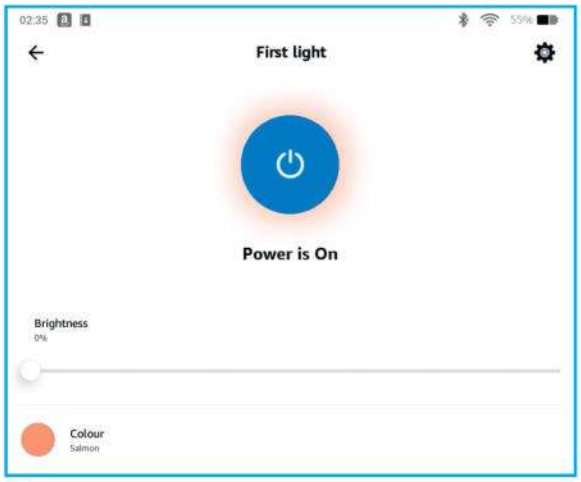
```
$ cd /etc/udev/rules.d
$ sudo wget https://raw.githubusercontent.com/
arduino/OpenOCD/master/contrib/60-openocd.
rules
$ sudo udevadm control --reload-rules
```

After that, deployment usually succeeds. The *Arduino IDE* confirms this by issuing the string `Programming Finished`, which – for inexplicable reasons – is displayed in red.

Be that as it may, the next step involves opening the Serial monitor and setting the bandwidth to 115,200. Finally, push reset to restart the provisioning process discussed above.

Kindle takeover

Once the provisioning information is visible, the device is ready for integration. For this, we return to the Kindle and open the *Amazon Alexa* application. After switching to the Devices tab, click the Plus symbol on the top-right corner of the screen to start the device activation option. Next, select the Add Device option. The shortcut list usually displays a Matter logo.



The Arduino Nano Matter Community Preview is ready for configuration.

Select the Matter option to continue the configuration process. In the first step, the program asks you whether the Matter logo is shown on the device – select Yes here and confirm that the device is powered on. After that, use the QR code scanner to scan the QR code found under the URL mentioned above. The security warning can be ignored – it is caused by the lack of certification.

After the app has confirmed a successful connection, the system allows you to assign the peripheral to the various groups Amazon implements in its smart-home system. When done, configuration can take place using the form.

At this point, the board is connected to the Alexa ecosystem. Should you want to break this connection for some reason, you need to run the sketch found at **https://bit.ly/lxf321man**. It performs a de-provisioning step and ensures that the board can be connected to another system.

Nothing really matters

When trying to get up to speed with Matter quickly, the Arduino Nano Matter Community Preview is among the cheapest (around €24) possibilities that you can use. If your application can live with the occasional warning about the device being uncertified, home-brew smart-home peripherals can now be created without you having to go through an expensive and effortful PCB layout cycle.

Furthermore, given that the Arduino board is based on a commonly used Silicon Labs processor, professionalising the solution is not particularly difficult because developers can reuse most of the source code from the prototype. The Arduino team also provides the board's schematics, thereby helping custom designs get off the ground quickly. LXF

» ACHIEVING CERTIFICATION OR NOT...

Removing the warnings about uncertified devices can only be achieved via Matter certification. The Matter protocol is governed by the Connectivity Standard Alliance (**https://csa-iot.org**), which acts as a gatekeeper for Matter peripherals – only if the product is white-listed in its list of IDs is it accepted by other systems.

Unlike the case of Bluetooth SIG certification (more information at **https://youtu.be/Mh4mDn2cFDo**), Matter certification is a technical boundary and is not purely legal. Full technical interoperability (aka removal of the warnings shown in custom projects) is achieved only after certification.

Achieving Matter certification is, sadly, a relatively expensive and multi-pronged process, which is best described by the flowchart from the Silicon Labs website: **https://bit.ly/lxf321flow**.

The situation is made expensive because Matter certification requires, at a minimum, being an Adopter Member of the CSA – the membership alone will set you back $7,000 per year, as per **https://csa-iot.org/become-member/**. In addition, the standardisation organisation behind the wireless protocol used also needs to agree – for this, additional membership and certification fees are required.

QUICK TIP

A Kindle is not mandatory! While this feature uses a Kindle Fire KFONWI as an Android device, other products can also be used. Download the *Amazon Alexa* application from your store of choice to transform a more-or-less random phone or tablet into a control terminal for smart-home devices living in the Amazon Alexa ecosystem.

DETOX

Credit: https://github.com/dharple/detox

Clean up filenames

You don't have to get up at 4am and practise yoga to find balance in life. **Shashank Sharma** found inner peace by cleaning up filenames.

OUR EXPERT

Shashank Sharma is a trial lawyer in Delhi, India. He's been writing about open source software for 20 years and lawyering for 10.

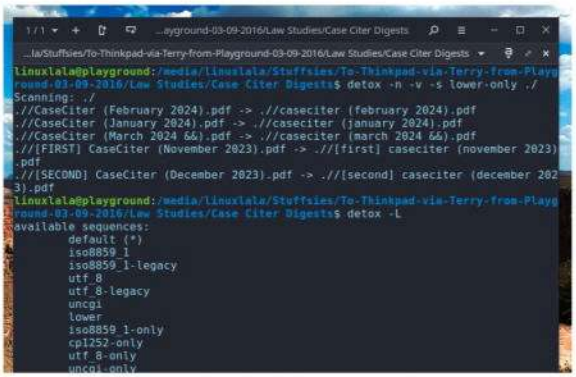
Filenames come in all shapes and sizes, and are often made up of unsavoury characters. That's because there's no defined rules for naming files, which are often a reflection of the user's mind. You'll find files with unnecessary spaces, hyphens, underscores, numbers and myriad other characters. If you've ever worked on a project that required sharing files with colleagues or peers, you'll have come across, suffered and resented this lack of file-naming convention. The lack of discipline in naming files can become even more tiresome when working on the CLI, as you have to escape some characters and spaces. Thankfully, *Detox* can help bring sanity to filenames.

Detox is equipped with a number of file-renaming rules, called sequences. You can define the sequence you wish to run on a specified file or files, or recursively on a directory if you wish to rename all comprising files. The **lower** sequence, for instance, replaces all upper-case characters in the filename with lower-case ones.

Can't fix what isn't broken

A detox of any kind only ever works if you first admit you have a problem. If you think *Detox* is not for you because you're disciplined and have a clear handle on your filenames, you're in denial. The good news is that help is always on hand and readily available.

In computing parlance, this means that *Detox* is available in the software repositories of many popular desktop distros. Depending on your distro, the version on offer might be a few releases old. Although this isn't a deal-breaker, *Detox* is under active development, and installing the latest version is fairly straightforward.



Not only is Detox easy to use, but the number of sequences on offer cover just about all manner of filename silliness.

Released under the BSD-3-clause licence, *Detox* has only a handful of dependencies, and you can easily satisfy these using the software repositories of your distro. Before installing *Detox* from source, make sure you have *autoconf, automake, gcc, bison, flex, make* and *pkg-config*. If you've used your current distro for any length of time, or have ever built packages from source, most of these dependencies will be installed. If not, head to the project's GitHub page (**https://github.com/dharple/detox**) for instructions on how to install

» DUPLICATE FILES

Along with keeping sane filenames, it's also important to prune your disks of duplicate files. But no one expects you to manually look through directories to identify possible copies of files. Thankfully, there are plenty of tools to perform just such an operation.

Fdupes works by comparing the MD5 signature of the files, followed by a byte-to-byte comparison to ensure all copies are identified. In addition to tracking down duplicates, you can use *Fdupes* to delete duplicate files, replace deleted files with links to the original, and so on.

You'll find *Fdupes* in the software repositories of most popular desktop distros, such as Ubuntu, Fedora, Arch and their derivatives. The `sudo apt install fdupes` command can be used to install the utility on Debian, Ubuntu and their derivative distros. You can similarly run the `sudo dnf install fdupes` command to install the utility on RPM-based distros.

At a minimum, *Fdupes* expects to be pointed to a directory to start hunting for duplicates. The command `fdupes <path/to/directory>` gets *Fdupes* to list all duplicate files it finds on the screen. You can use the **-r** command option to make *Fdupes* recursively look through directories. You must use the **-d///** command option if you wish for *Fdupes* to delete the duplicate files. The utility still asks you for confirmation before deleting any files.

these dependencies on distros such as Ubuntu, Debian and their derivatives, Mac OS, FreeBSD and so on.

That done, you can build *Detox* from source. First, grab the latest release tarball from the project's GitHub page, then extract the files and execute `./configure`, `make` and `sudo make install` to install *Detox*.

Detox your files

Before you perform any renaming operations with *Detox*, it's important to test it first. Thankfully, it offers a dry-run option, which shows the renaming operation *Detox* will perform, without changing the files.

The `detox ./ -n` command makes *Detox* check all files in the current directory and list on the screen how files with troubling names will be renamed.

```
$ detox ./ -n
.//Inspection Supertech.pdf -> .//Inspection_Supertech.pdf
.//Supertech mail.pdf -> .//Supertech_mail.pdf
.//IIFL Finance vs AVJ Developers.pdf -> .//IIFL_Finance_vs_AVJ_Developers.pdf
.//Mayank's health notes-1.pdf -> .//Mayank_s_health_notes-1.pdf
.//ORDER -06.07.2021.pdf -> .//ORDER-06.07.2021.pdf
```

The first filename in each instance above is the troubling one discovered by *Detox*. The name following the arrow (->) is how *Detox* offers to rename it. *Detox* offers to replace all spaces and apostrophes with an underscore (_), but hyphens (-) are left untouched.

This is the default sequence, but if we run `detox -s lower ./ -n`, *Detox* still replaces spaces and other characters with underscores, but additionally changes upper-case letters to lower-case:

```
$ detox -s lower ./ -n
.//Nilkamal-office-chairs.pdf -> .//nilkamal-office-chairs.pdf
.//Delhi High Court Affidavit.pdf -> .//delhi_high_court_affidavit.pdf
.//Annexures_Part_2.zip -> .//annexures_part_2.zip
.//ORDER-22.02.2021.pdf -> .//order-22.02.2021.pdf
.//APPL_SUMMONING_WITNESS.doc -> .//appl_summoning_witness.doc
```

When ready to let *Detox* rename files, drop the **-n** command option and execute the *Detox* command.

For a list of all available sequences, run `detox -L`:

```
available sequences:
        default (*)
        iso8859_1
        iso8859_1-legacy
        utf_8
        utf_8-legacy
        uncgi
        lower
        iso8859_1-only
        cp1252-only
        utf_8-only
        uncgi-only
        lower-only
```

Let's say you have filenames with upper-case letters as well as characters such as parentheses or brackets. If you want to change the upper-case letters to lower case, but wish to retain the other special characters instead of having them replaced with an underscore or hyphen, you can use **lower-only**.

```
$ detox -n -v -s lower ./
.//CaseCiter (February 2024).pdf -> .//caseciter-february_2024.pdf
.//CaseCiter (January 2024).pdf -> .//caseciter-january_2024.pdf
.//CaseCiter (March 2024 &&).pdf -> .//caseciter-march_2024_and_and.pdf
.//[FIRST] CaseCiter (November 2023).pdf -> .//first-caseciter-november_2023.pdf
.//[SECOND] CaseCiter (December 2023).pdf -> .//second-caseciter-december_2023.pdf
```

Above, *Detox* offers to change upper-case letters to lower case, and replaces square brackets and parentheses with underscore. Compare that with the output of the **lower-only** sequence:

```
detox -n -v -s lower-only ./
.//CaseCiter (February 2024).pdf -> .//caseciter (february 2024).pdf
.//CaseCiter (January 2024).pdf -> .//caseciter (january 2024).pdf
.//CaseCiter (March 2024).pdf -> .//caseciter (march 2024).pdf
.//[FIRST] CaseCiter (November 2023).pdf -> .//[first] caseciter (november 2023).pdf
.//[SECOND] CaseCiter (December 2023).pdf -> .//[second] caseciter (december 2023).pdf
```

Not only does *Detox* leave the characters as is, even the spaces remain untouched, and only the upper-case letters are affected.

Not for everyone

CLI purists would scoff at the idea of using a dedicated utility to perform such an operation, arguing that *sed* is more than up to the task of renaming files.

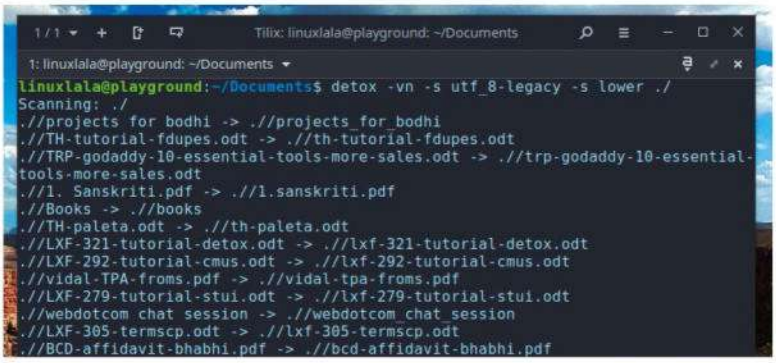
For instance, you could run `for i in *' '*; do mv "$i" `echo $i | sed -e 's/ //g'`; done` to remove all single spaces from the filenames in the current directory. If you wanted to replace the single space with an underscore, you could run `for i in *' '*; do mv "$i" `echo $i | sed -e 's/ /_/g'`; done`. If you wanted to retain the letters, numbers, periods and underscores, but replace all other characters with an underscore, you could similarly run `for i in *' '*; do mv "$i" `echo $i | sed -e 's/[^A-Za-z0-9._]/_/g'`; done`.

If you've never used *sed* before, please copy a few files into a test directory before you start experimenting, lest you lose the files for ever. LXF

QUICK TIP

People are often quick to delete the source directories from where they install apps and utilities. Doing so deprives you of the option to properly uninstall these programs. To remove *Detox* from your system, run the `make uninstall` command from *Detox's* source directory from which you executed the `sudo make install` command.

Depending on your filename mania, you might want to combine sequences for optimum results. Make sure to do a dry run first.



LINUX BASICS

Secure the Linux system and its files

Part Seven! Don't miss next issue, subscribe on page 16!

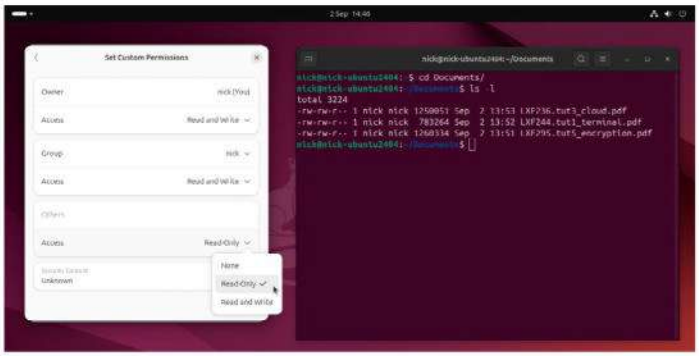
The ever-watchful **Nick Peers** discovers how Linux is built from the ground up to help keep you and your data secure.

OUR EXPERT

Nick Peers has seen his fair share of viruses over the years, but has yet to see any appear on his Linux machine. Talk about tempting fate...

One key advantage of Linux is that it's been built with security in mind. As earlier versions of Windows practically invited malware on to their insecure systems, Linux's early developers were taking notes and making sure they didn't fall into the same trap. As a result, the Linux kernel – the core interface between your PC's hardware and its own processes – is written with security as a key priority.

In this *Linux Basics* tutorial, we're exploring the most notable ways in which Linux is designed to keep everything from your user account to your files safe, snug and secure.



You can view a file's permissions using both your file manager (left) and the 'ls -l' command in the terminal (right).

User privileges

One of the most visible ways in which Linux is more secure than the likes of Windows is with its user privilege model. They say imitation is the sincerest form of flattery, so when Windows introduced its User Account Control dialog in Windows Vista – a desultory attempt to block malware through a confirmation prompt that requires you to simply click Yes or OK to any process's attempt to gain carte blanche access to your system – many Linux users sat back and smirked.

Even now, the Windows UAC dialog fails to mask its underlying insecurity, where everyone is given admin levels of access by default. In contrast, Linux applies a strict user privilege model whereby even administrator-level users don't have full access to the system. Instead, the highest level of access is restricted to a single root user (also known as superuser). Administrators can gain temporary access

QUICK TIP

Password security good practice involves changing your passwords on a regular basis – including the one you use to log into Linux. Set yourself an alarm to change this every six months – visit Settings > System > Users to do so in Ubuntu.

» HARDWARE SECURITY

The Linux filesystem is designed to allow Linux to treat everything as a file – even hardware devices. Type `sudo lshw` and press Enter, and you're given a long list of all the internal and external hardware devices connected to your PC. Each one comes with an entry marked 'logical name'; you'll see it's a path that begins with **/dev/**.

Type `cd dev` into the terminal followed by `ls -l` and you'll see a long list of entries, complete with permissions, owner and group – just like with regular files and folders. These permissions determine who has access to these devices. Look closely at the groups and you'll see entries like 'video', 'dialout' and 'disk'.

Now type `id` and press Enter – this reveals that your user is a member of many of these groups, including audio, video and render (for video rendering). This means your user has access to these devices through its membership of those groups.

In most cases, you need do nothing more – you have all the access you need to the hardware on your PC, which you naturally control using the infinitely more user-friendly *Settings* app on your desktop. Only on rare occasions might you need to grant access to a user in order to give it access to hardware devices – for example, to enable hardware acceleration support for the *Jellyfin* (**https://jellyfin.org**) media server, you'll need to add the jellyfin user to the render group:

```
$ sudo usermod -a -G render jellyfin
```

to the root user one of two ways. First, when you're logged on to your desktop and you – or the system – attempts to perform a task that requires superuser levels of access, you see a prompt requiring you to enter your user password. Once entered, your user gains temporary root access to the system for the purposes of performing that specific task only.

Second, when you're logged on to a terminal, you gain root access one of two ways. On Ubuntu-based systems, you prefix commands using `sudo`. Again, you are prompted for your user password the first time you use *sudo*, and then you can continue entering *sudo* commands for a short period of time (five minutes by default) before you need to enter your password again. On other systems like Debian, *sudo* isn't installed by default. Instead, you'll have been prompted during setup to create a separate password for the root user. You need to explicitly switch to the root user in a terminal session (type `su` and press Enter, then enter the root user's password) to perform administrative tasks.

Secure filesystem

Linux's default filesystem – ext4 – is another example of where security plays a key role. Every single file or folder is assigned an owner and a set of permissions that restrict access. There are three types of permission that can be granted to a file: read access (**r**), write access (**w**) and execute access (**x**), the latter required for programs or scripts to be able to run. A file can be assigned none (**---**), some (**r--** for read-only or **r-x** for read and execute) or all (**rwx**) of these.

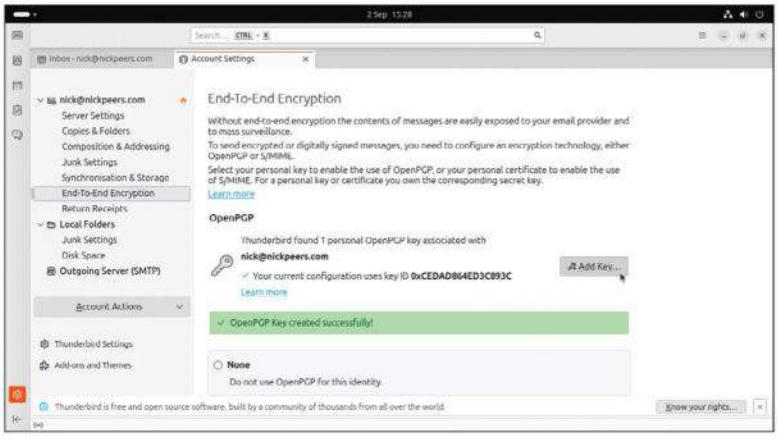
Folders work in a similar way, but their permissions also affect the files inside. For example, a user may have read access to a file, but unless they also have read and execute access to its parent folder, they can't actually navigate to where the file is to open it. Users also need write access to a folder to create or delete files inside folders.

What make ext4 even more secure is that permissions are assigned to three different levels: a single user deemed the file or folder's owner, a specified user group, and others (basically everyone else). To view a file's current permissions – along with its owner – open your file manager *(Files* in Ubuntu) and choose Properties. You'll see its permissions as they apply to your user account under Permissions – click this to view more information about the file. You'll see who the owner is (typically you if you're examining files in your **Home** folder) and their permissions, plus the assigned group (often the same name as the owner) and everyone else.

If you want to view a file's permissions in the terminal, navigate to the folder it's in, type `ls -l` and press Enter. Look to the left of the file or folder where you'll see something like `-rw-r--r--` next to a file, and `drwxr-xr-x` next to a folder. Ignoring the leading character (`-` for a file, `d` to denote a folder or 'directory') leaves you with three sets of `rwx`, which denote the permissions for the owner, group and others.

Changing permissions

Only the file owner or root (via *sudo* in the terminal) can change a file or folder's permissions. This is a potentially dangerous operation and if you're not careful, you could lock yourself out of your system and effectively screw up your installation. As a result, avoid touching any files outside of your personal **home** folder (or ~ in the terminal). If you're planning to experiment, work on copies of the original file so it's left untouched.



Thunderbird supports both OpenPGP and S/MIME to allow you to digitally sign and encrypt your email communications.

With these safety barriers in place, you can change the permissions of any files you own through your file manager. Here, it's a simple case of using the drop-down menus on the Permissions tab in Ubuntu or checking boxes, as is the case with Mint's file manager.

If you're in the terminal, familiarise yourself with the *chmod* command, which is used to change file and folder permissions. One way to use *chmod* is to add or remove specific permissions for the specified user class: owner, group or others. This is done using options such as **+r** or **-w** but this approach can be fiddly to implement.

A quicker way is to redefine the permissions of a specific file or folder for the file's owner, specified group and everyone else with a single command. For this to work, you need to define each class's permissions as a single number between 0 and 7. This number is calculated by adding up the permissions you're granting: 4 (read), 2 (write), 1 (execute) and 0 (no permissions). You then add these together to give the specific set of permissions for the defined user – for example, 6 (4+2) equates to read/write, while 7 (4+2+1) represents full access. So, to give all three classes full access (NB do not do 777 access on production machines) to a file, you'd use the following command:

```
$ sudo chmod 777 file.txt
```

More sensible is to give the owner full access, while restricting everyone else (including the specified group) to read and execute rights only:

```
$ sudo chmod 755 file.txt
```

More common numbers include 770 (full access for the owner and group, but no access to anyone else) and 644 (read/write access to the owner, read-only access to the group and everyone else). If you're struggling to work out what numbers you need to assign, visit **https://chmod-calculator.com**, where you can check the read, write and execute boxes for each of the three classes to come up with your final three-digit number.

By default, permissions applied to a folder apply to that folder only – its contents are left as they are. If you wanted to apply the same set of permissions to a folder and all its contents, you'd need to use the

QUICK TIP

When setting up new users on your system, avoid giving too many users Administrator status. Navigate to Settings > System > Users to set up new users – leave the Administrator switch off to ensure they can't do certain tasks like add/remove users or uninstall software.

QUICK TIP

Visit https://itsfoss.com/gpg-encrypt-files-basic/ for a guide to encrypting files for sharing with others as well as encrypting and decrypting individual files on the fly using your own PGP private and public keys.

recursive (**-R**) flag, like so:

```
$ sudo chmod -R 644 folder
```

Network permissions

Changing permissions isn't always the best way to gain access to a file. If you grant wider access to the others group because you need access to a file you don't own, you're effectively giving all users and processes access to that file. First, ask yourself if you need permanent access to the file. If, for example, you simply need to edit a system configuration file, just use the *sudo* command to access it in a terminal window to give yourself temporary root-level access to the file, such as when editing network permissions:

```
$ sudo nano /etc/samba/smb.conf
```

If you genuinely need permanent access to the file via a specific user, you have three options: change the file's owner to the user who needs access, add the user to the group assigned to the file, or change the file's group. Which option you choose depends on your circumstances – changing a file's owner or group could have repercussions elsewhere, particularly if they're currently assigned to root. Usually, adding the user to the file's designated group is the safest option; either way, Linux allows you to do both via the terminal.

First, identify the file's current owner and group. Again, `ls -l` reveals this information (the owner and group are listed next to the permissions). If the current owner and group are root, then in most cases, you'll want to leave this alone. If the group is someone else, try the following command, substituting `group` and `user` with your target group and username:

```
$ sudo usermod -a -G group user
```

Verify your user is now a part of that group by typing `id user` (again, substitute `user` with your username). Users can be members of more than one group at a time – the *id* command should confirm this. Whatever you do, don't omit the **-a** flag – doing so would remove the user from all groups other than the one you've just added it to.

Change ownership

Should you want to change the owner and/or group assigned to a file or folder, you'll need to use the *chown* command. Again, this is a potentially dangerous move, so don't use it on anything outside your **Home** folder. As before, it's issued from the terminal:

```
$ sudo chown <option> file.txt
```

The `<option>` section determines the name of the new owner and/or group. To change the owner while leaving the group as is, use `sudo chown user file.txt`, replacing `user` with your target username. To change the group, use `sudo chown :group file.txt` instead, again changing `group` to the name of your chosen group. To change both, type `user:group` (or just `user:` if you want both user and group to be the same).

As with *chmod*, the **-R** recursive flag is also available to allow you to change the owner and/or group of a folder and all its files, like so:

```
$ sudo chown -R user:group folder
```

Disk encryption

Your Linux distro is also capable of providing security for individual files and folders through encryption. Full-disk encryption is only possible when first installing your distro from scratch – for example, in Ubuntu when you come to choose Installation Type, click the Advanced Features button and make sure Use LVM is selected and Encrypt The New Ubuntu Installation is ticked before clicking OK followed by Install Now.

You need to provide a strong password or passphrase as your security key – this needs to be entered every time you boot your computer to unlock the drive – and there's an option for a recovery key in case you forget the security key. In either event, we recommend storing these passphrases somewhere secure, like your password manager.

Full disk encryption is only really needed should you worry about your computer being stolen – so for obvious reasons, it's a better choice for laptops than desktops, for example. But there may be times when you need to protect individual volumes (such as a USB flash drive) – *see the Encrypt Volumes box, right, for*

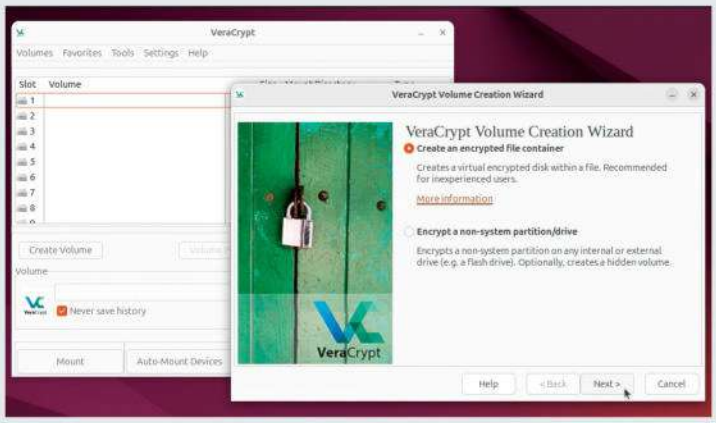
» ENCRYPT VOLUMES

Another use for encryption is to provide yourself with a secure means of transporting files on external drives. Linux has encryption tools – *DM-Crypt* – within the Linux kernel, so if you're comfortable with command-line usage, `sudo apt install cryptsetup-bin` will install *Cryptsetup*, which makes use of *DM-Crypt*.

For those who want a simpler life, the best tool for any form of encryption is *VeraCrypt* (**https://veracrypt.fr**), which you can install after downloading the appropriate DEB package via its Downloads page. Once installed, open the app and click Create Volume. You're given two choices: encrypted file container is the safest, because it can be added to any drive without destroying the files already on there, but you can also format an entire partition or drive so all of its contents are encrypted.

One obvious use for *VeraCrypt* is to provide you with a secure means of storing data on vulnerable portable devices, such as USB flash drives. The only drawback of this approach is that *VeraCrypt* must be installed on any computer you plug the flash drive into, so you can open and decrypt the volume to get at its data.

You'll find comprehensive documentation on the *VeraCrypt* website (**https://veracrypt.fr/en/Documentation.html**), including a handy step-by-step guide that covers the basics of setting up and accessing your encrypted volumes. If you have particularly sensitive data, you can go one step further and store it in a hidden volume, which is stored inside a standard *VeraCrypt* volume, making it even harder to detect.



VeraCrypt's step-by-step wizard takes you through every part of the process, with easy access to useful information along the way.

details – or files that you need to share over insecure platforms like email.

Encrypt files and emails

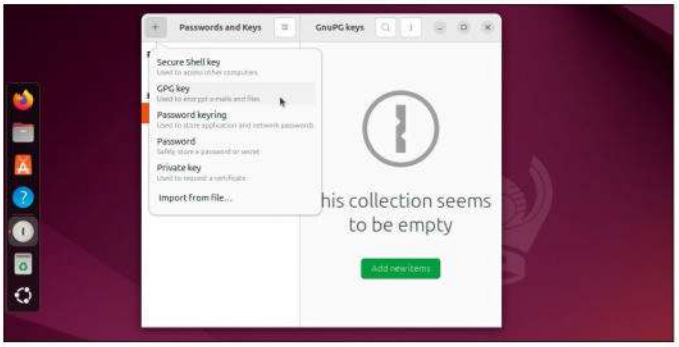
When it comes to file encryption, the most popular tool – largely because it's standard in most distros – is *GnuPG*. It works using asymmetric encryption, which means it requires a pair of keys – a public key you share with others, and a private key that remains private to you. The public key is used by others to encrypt files, so they can be shared safely using whatever medium they wish (including insecure platforms like email). The private key is then used by you to decrypt the file.

To verify *GnuPG* is installed on your system, open a terminal, type `gpg --version` and press Enter. You'll see its version number as well as its supported algorithms. Note, *GnuPG 2.4* and later use the industry standard AES256 cipher as the default. Although *GnuPG* is designed to work from the command line, the step-by-step guide *(below)* reveals an easier way to create and manage keys using Ubuntu's *Passwords and Keys* app.

From here, you can encrypt and decrypt files using PGP from the command line. You'd usually do this for the purposes of sharing sensitive files with others, so you need the public key of the person you're sending a file to so you can encrypt the file for them to decrypt.

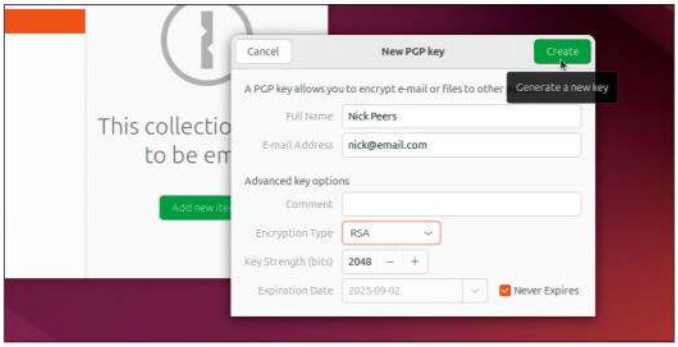
You can also create keys within apps, including email for encrypting your communications. Here, the most obvious example is *Thunderbird,* which comes with PGP support built in. Visit **https://mzl.la/4dND4xR** for a complete guide to doing so, but simply put, you need to visit your account settings page and select End-To-End Encryption, then click Add Key to create a new key from scratch or import your PGP key created using *Passwords and Keys.* Once configured, you can digitally sign (to prove your identity) or encrypt messages directly from the email composition window using the buttons provided. LXF

CREATE A PGP KEY FOR FILE ENCRYPTION



1 Open Passwords and Keys

Open the Dash, click on Utilities and then click Passwords And Keys to open the app. A window opens displaying any passwords, certificates, keys and other items – if you have none of these, a 'This collection seems to be empty' message is displayed. Start by clicking the + button in the top-left corner and choosing GPG Key from the drop-down.

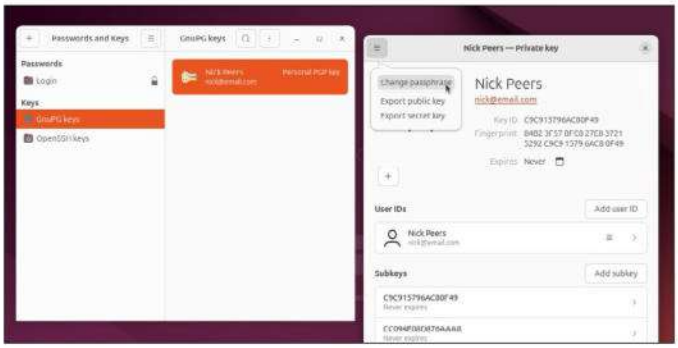


2 Create public key

Fill in your name and email address before looking at the Advanced Key Options section. The default encryption type (RSA) and key strength (2,048 bits) should be sufficient. By default, this key will never expire – untick this and set an expiry date if you wish it to end after a set period. Click Create to move on to the next step.

3 Set password

If prompted, click Deny when asked to allow inhibiting shortcuts. You are now prompted to create a password that will be needed to decrypt any files you encrypt through PGP going forward. This should be long (minimum 12 characters) and strong, but either memorable or stored somewhere safe like your password manager. Click OK when done.



4 Generate your private key

A dialog pops up asking you to perform lots of random movements with your mouse to help generate the key. Once complete, you'll see your Personal PGP key appear in the list, ready for use. Double-click it to reveal its properties and change the passphrase, or right-click the entry if you want to export it as a file or to delete it.

INSPIRCD

Credit: www.inspircd.org

IRC: the oldest chat system going on strong

Chatterbox **Nate Drake** walks you through how to set up and secure your very own IRC server, as well as master its many commands.

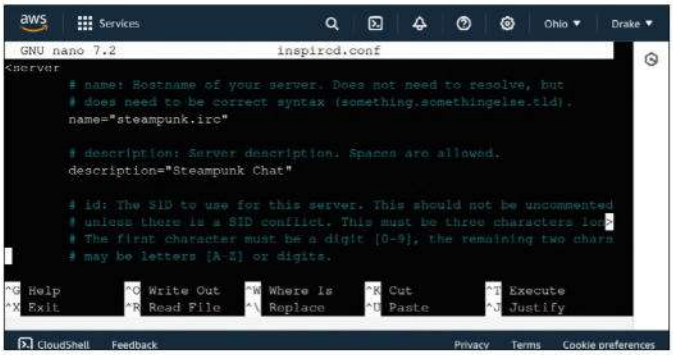
OUR EXPERT

Nate Drake is a tech journalist specialising in cybersecurity and retro games. His first attempt to join an IRC ethical hacking channel landed him in a fan group for 'furries' instead.

While IRC has been dwarfed by instant messaging programs since its inception in 1988, it still remains popular with groups like tech enthusiasts, perhaps due to the protocol's open nature and the fact there's a number of open source, customisable clients.

The fact that said clients aren't as easy to set up as, say, *Signal* messenger is kind of the point. When it comes to both exclusivity and geekiness, IRC leaves other programs and protocols in the dust. On the plus side, this usually means you'll find channels of very well-informed people on a number of obscure topics. When researching this article, Nate went down a rabbit hole on an IRC channel devoted to llamas, for instance.

In this guide, you'll learn how to set up and configure your very own IRC server, secure it using TLS or the dark net, then help others to connect.



Your **IRC** server settings are managed from the inspircd.conf configuration file. Restart the service each time you make changes.

Getting started

In order to operate an IRC service, you need to have access to a server. Technically you can do this using your home machine, but it would consume a large amount of system resources, not to mention require a large amount of port forwarding on your router.

VPS (virtual private servers) are ideal for running IRC. We used the free tier of an Amazon EC2 instance running Ubuntu 24.04 (**https://aws.amazon.com**) – be aware that charges can apply (ensure you disable your server when not used) and we're not responsible! We've also chosen *InspIRCd* as our server software, given that it's relatively easy to set up.

Technically you can install the existing version (currently 3.17) in Ubuntu's repositories but if you do this, *InspIRCd* won't automatically enable third-party modules, such as those necessary for SSL support. The version in the repos is also rather dated. To install the latest version, you need to compile the code from source. First install the necessary dependencies:

```
$ sudo apt install git perl g++ make gnutls-bin
libgnutls28-dev pkg-config
```

Download the latest version (currently 4.2.0):

```
$ wget https://github.com/inspircd/inspircd/archive/
refs/tags/v4.2.0.tar.gz
```

Extract the archive – for example, `tar xvf ./v4.2.0.tar.gz` – then use `cd` to switch to the new directory. In our case, this was `cd inspircd-4.2.0`. To compile *InspIRCd* with SSL support, you need to run:

```
$ perl ./configure --enable-extras ssl_gnutls
```

Next move the files to the correct location with:

```
$ make -j2 install
```

You now need to make a copy of one of the example configuration files to work with:

```
$ cp /usr/local/src/inspircd/run/conf/examples/
inspird.example.conf /usr/local/src/inspircd/runconf/
inspircd.conf
```

The file is laid out very logically, so it's just a question of going through each section and changing the default values to something more meaningful.

First, find the **server** tag and change the server name. This doesn't have to be a formatted as a web address – for example, steampunk.irc. Next, change the description for your IRC server to something more exciting, such as Steampunk Chat.

Find the **admin** tag and enter some appropriate details – for instance:

```
<admin name="Nate Drake"
  nick="nate-drake"
  email="nd@natedrake.com">
```

As with the server name, the email address doesn't have to be real but must be correctly formatted.

Now find the **bind** tag and remove 127.0.0.1 from the Address field. This allows outside connections.

QUICK TIP

An example *InspIRCd* configuration file is available from https://bit.ly/inspircdconf. Take some time to read through this before you begin editing your own. If you compile *InspIRCd* from source, more examples are available in /inspircd/run/conf/examples.

Finally, find the **oper** tag. From here, you can enter your oper(ator) name and a password. By default, the **host** value only allows logins from local domains but you can amend this by changing **localhost** to **t**:

```
<oper name="nate"
    password="greensleeves"
    host="*@*"
    type="NetAdmin"
    maxchans="60">
```

If you compiled *InspIRCd*, the **oper** tag doesn't seem to be in the example configuration but you can type in the values yourself. Just make sure also to define the **Netadmin** type beforehand – for example:

```
<type name="NetAdmin"
    class="ServerOperators"
    classes="BanControl HostCloak OperChat
SACommands ServerLink Shutdown"
    vhost="steampunk.irc">
```

You also need to define the **classes** for the type of commands supported by the operator. You can find an example of this on lines 72- 76 at **https://github.com/inspircd/inspircd-docker/blob/master/conf/opers.sh**/. Press Ctrl+X, Y, then Enter to save your changes.

Mixed messages

The MOTD (message of the day) appears each time an IRC user connects to the server. It can also be invoked at any time with `/motd`. It typically includes a description of the server, the network on which it's running, and a list of rules to follow. To get started, run:

```
$ sudo nano /etc/inspircd/inspircd.motd
```

You can now type in your message for users. IRC admins often use ASCII art websites to prepare some presentable text. But as Bob Ross said, this is "your world", so feel free to compose a message as you see fit. When you're done, exit in the same way as for the configuration file.

Start your server

InspIRCd configuration uses port 6667 by default. Open this port by running:

```
$ sudo ufw allow 6667
```

If you followed our recommendation to run your IRC server in an AWS EC2 instance before, you also need to open this port from the main console's security group settings.

Pull up the list of running instances on the dashboard, then click into the Security tab. Next, select the link under Security Groups. On the next page, choose Edit Inbound Rules, then under Type select Custom TCP. You can now enter 6667 under Port Range. Click Apply Changes to save.

Once your ports are open return to the CLI and run:

```
$ sudo systemctl restart inspircd
```

Double-check everything's running correctly with:

```
$ sudo service inspircd status
```

Test your IRC server

Once your IRC server seems to be functioning correctly, it's time to test it out with a dedicated client.

There's a number of amazing graphical programs for using IRC, such as Gnome's own *Polari,* but for testing purposes, just fire up a terminal on another Ubuntu machine and install the CLI IRC client *Irssi* with:

```
$ sudo apt install irssi
```

The **MOTD** usually contains network information and rules. Many operators make use of **ASCII art** to add a touch of flair.

On first run, you can connect to the IRC server using its public IPV4 address – for example:

```
/connect 3.144.174.122
```

By default, *Irssi* simply uses your account username as your IRC nickname. You can change this via the `/nick` command – for example:

```
/nick steampunk-nate
```

The MOTD should now appear as well. To get started, you need to join an IRC channel via the `/join` command. Because this is a new IRC server, doing this creates a channel – for example:

```
/join #steampunkchat
```

If you want to message someone privately, just use the `/msg` command:

```
/msg alice Dieselpunk is great but not as good as
Steampunk. :-)
```

You can use the commands `/leave` and `/disconnect` to leave a channel or close the session altogether.

Smooth operator

Whilst logged in to IRC, you can use the `/oper` command to elevate your privileges at any time. Using the example configuration above, for instance, type:

```
/oper nate greensleeves
```

Now imagine, that IRC user alice has made a gauche remark about preferring the dieselpunk genre, so you decide to boot her out of the steampunk chat channel:

```
/kick steampunkchat alice
```

If alice commits a greater infraction, you can disconnect her altogether:

```
/kill alice
```

Operators can usually view the IP address from which users connect and ban them on this basis. This caused an issue for Nate when connecting to IRC via a shared VPN server. If this is happening to you repeatedly, we suggest getting a dedicated IP address or connecting via Tor (see below).

Secret sessions

By default, any user can join any IRC channel. To see all available channels once connected, run `/list`.

As we've learned, you can chat to individual users with the `/msg` command:

```
/msg Alice I still say Steampunk is the best Sci-Fi
genre!
```

If you want to have a private conversation with multiple people you can also set up a secret channel. To get started, create a new channel by using `/join`:

```
/join #shadow
```

QUICK TIP

Different channels have various modes to allow or permit certain activity – for example, to ban certain users or to make channels invitation-only. To check active modes on a specific channel, just run `mode #channel name`.

QUICK TIP

Another good reason to run your IRC server through Tor is that it will work out of the box with the version of *InspIRCd* in Ubuntu repositories. This saves you the trouble of compiling and building from source.

Next, make your IRC channel secret so it doesn't show up when users enter `/list`:

```
/mode #shadow +s
```

You can also add the letter **i** to this command to make chat invite-only. This means a user can only join if someone already in the channel allows it:

```
/invite alice #shadow
```

During our tests for this article, we discovered that if a user leaves an invite-only channel, an existing user must reinvite them if they want to reconnect. Use the **i** flag at your own discretion.

You can also secure the channel further by adding a key. This effectively functions as a password that users need to enter in order to join – for example:

```
/mode #shadow +k redalert
```

Others can join the channel by using the relevant command and the password:

```
/join #shadow redalert
```

Talking TLS

By default, IRC conversations between the client and server are unencrypted. In other words, anyone monitoring your traffic can read your conversations. It has been one of the turn offs for it in the past.

If you compiled *InspIRCd,* you can configure it to use TLS. To get started, create a new folder to store your SSL certificate in the main **inspircd** directory and switch into it using: `mkdir cert && cd cert`.

If you're lucky enough to be running *InspIRCd* on your own web server complete with SSL certificate, at this stage you only need to download said certificate to this folder. Similarly, if you have a web server with a dedicated domain, you can obtain a free certificate from **https://letsencrypt.org**.

If, like us, you are running a VPS with only a public IP address, you can generate and sign your own SSL certificate. This provides equivalent security to one validated by a certificate authority but will throw up warnings for any users trying to connect. Fortunately, self-signed certificates are quite common in the world of IRC.

Generate your SSL certificate and key with:

```
$ certtool --generate-privkey --outfile key.pem
$ certtool --generate-self-signed --load-privkey key.
pem --outfile cert.pem
```

During the generation process, you're asked a series of questions. Pressing Enter to provide the default answer works in most cases. When asked if the certificate will be used for signing, make sure to choose Y. You also need to specify the number of days for which the certificate will be valid.

Next, return to editing your **inspircd.conf** file. Use the # to comment out the existing **bind** tag and replace with:

```
<bind address="*"
    port="6697"
    sslprofile="Clients"
    type="clients">
```

This instructs *InspIRCd* to use port 6697, which is the default for secure connections over IRC. Below this, insert this line to enable the GNUTLS module:

```
<module name="m_ssl_gnutls.so">
```

Finally, add the sslprofile for client connections. The keyfile and certfile values should reflect the actual location of the **cert** folder you created earlier:

```
<sslprofile name="Clients"
        provider="gnutls"
        cafile=""
        certfile="/etc/inspircd/cert/cert.pem"
        crlfile=""
        dhfile=""
        hash="sha3-256"
        keyfile="/etc/inspircd/cert/key.pem"
        mindhbits="1024"
        outrecsize="2048"
        priority="NORMAL"
        requestclientcert="yes"
        strictpriority="no">
```

Once this is done, save and exit. Make sure you enable your chosen port for secure IRC connections via `ufw`.

If you're running an EC2 instance like us, you also need to open this port in the security group settings as outlined above. Now restart *InspIRCd* to apply your changes:

```
$ sudo systemctl restart inspircd
```

You now need to make some changes to your IRC clients. The steps vary depending on your chosen program. For instance, *Polari* has a rocker switch labelled Use Secure Connection. KDE's *Konversation*

» SETTING PERMANENT CHANNELS

The default IRC configuration is that virtually any user can create their own channel using `/join` but it will wink out of existence as soon as the last person leaves. This can be a nuisance to set up again if you're creating a secret or invite-only channel.

InspIRCd's permchannels module is designed to segue around this issue. According to the official documentation, if you create a channel using `/mode` with the **p** flag, *InspIRCd* automatically creates a permchannels configuration file. However, the easiest way to get started is just to define the channel in the configuration file itself.

To get started, use *nano* or other favourite utility to edit the **inspircd.conf** file.

First you need to invoke the permchannels module. Insert this line after the 'connect' section:

```
<module
name="permchannels">
```

Now you can define your permanent channel. For instance, to set up a dieselpunk channel, insert:

```
<permchannels
channel="#dieselpunkchat"
        modes="+bnt *!*@
baduser"
        topic="Welcome to
Dieselpunk chat!"
        topicsetby="Nate"
        topicts="956188800"
        ts="726192000">
```

In the case of modes, we've used the **b** (ban) flag to stop anyone with the name baduser from joining. The **n** flag (noextmsg) ensures that users outside the channel can't message it. The **t** (topiclock) prevents non-channel users from changing the channel topic.

If you wish to create a secret group, add **s** to the modes. You can also put in an **i** to make it invite-only. For a full list of channel modes, visit **https://docs.inspircd.org/3/channel-modes/**.

» NAVIGATING THE CHANNELS

If this is your first time using IRC, you need to choose an IRC client. We've explored a few of these in the main part of the tutorial.

As a rule, when you first connect to a server, you should register your current nickname to stop others taking it. You can also add a password so only you can log in, plus an email address in case you need to reset:

```
/msg NickServ REGISTER password123 nd@natedrake.com
```

If you've previously registered with this IRC server, you can log back in by providing the password you set earlier:

```
/msg NickServ IDENTIFY nate_ password123
```

Once this is done you can use `/join` to enter an existing channel or create a new one.

If you're bored of spartan black and white text, you can type character codes into the window to change colour. However using Ctrl+K in most IRC clients enables you to select from a pop-up palette.

Technically, you can enter `/say` before speaking but the usual convention for ordinary text is to just type and press Enter. As conversations can become very laborious, most users put your username when talking to you in the main chat, for example:

```
[18:46] <nate_> alice: Welcome to #steampunkchat.
```

If a user is performing a particular action, they can also use `/me`. For instance typing `/me is learning how to use IRC` appears in the channel as: '(Your name) is learning how to use IRC'.

IRC client similarly has a checkbox for Use SSL. Make sure the client also uses the new port number.

If you're using a self-signed certificate, the client may show you a warning. For instance, *Konversation* asks if you want to trust the certificate for that session or for ever. During our tests, we were unable to establish a secure connection when attempting to use *Polari* with the IRC server with a self-signed certificate. If readers are able to get this working, please don't hesitate to get in touch.

Going dark

Using TLS is an excellent way to prevent snooping on your conversations but the IP address of your server and other users are still visible to bad actors, who can target you for online abuse.

One simple solution is to set up your IRC server as a Tor hidden service. As data is encrypted and routed through multiple relays, it's almost impossible for users to know your server IP. They also have the peace of mind of knowing their location can't be traced either.

To get started, you first need to install Tor on your *InspIRCd* server:

```
$ sudo apt install tor
```

Next, use *nano* to edit the **<bind>** tag in the **inspirdc.conf** file:

```
<bind address="127.0.0.1" port="6667" type="clients">
```

As connections will be routed through Tor, only the home address is required.

Now use *nano* to edit the Tor configuration file:

```
$ sudo nano /etc/tor/torrc
```

Scroll down to the section marked 'This section is just for location-hidden services'. Uncomment the following two lines:

```
HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 127.0.0.1:80
```

Change both sets of 80 to your IRC port number:

```
HiddenServicePort 6667 127.0.0.1:6667
```

Save and exit, then restart *InspIRCd* as outlined above. Type `sudo killall tor && sudo tor`. Tor generates an .onion address for your IRC server. View this:

```
$ sudo cat /var/lib/tor/hidden_service/hostname
```

You can now configure your IRC client to access the server. How you do this will vary depending on your particular client but in all cases you must install and launch Tor first.

You then need to enter your client's network settings and enable a proxy. The address should be 127.0.01 and the port 9050. The proxy type is SOCKS5. After this you can use the .onion address as the server address. The port number should be the same as you set in the **bind** tag earlier – such as 6667.

To check everything's working correctly once you're connected, run the `/who` command. Your IP address should appear as 127.0.0.1.

Further reading

As we've learned, IRC is incredibly customisable, which explains its enduring popularity even in light of similar alternatives like Discord.

If you want to expand *InspIRCd* further, we suggest reading through the documents section of the support pages (**https://docs.inspircd.org**). It was here we learned, for instance, that in order to ensure the program launches each time the server reboots, you only need to run:

```
$ sudo systemctl enable inspircd.service
```

The site FAQ also states that *InspIRCd* is sometimes mistaken for malware by antivirus programs, given that hackers traditionally used IRC to control botnets. The website suggests reporting this false positive to the developers to avoid seeing any alerts. LXF

Using a self-signed SSL certificate can trigger alerts in IRC clients but users can authorise them for that session or permanently.

BACK ISSUES » MISSED ONE?

ISSUE 320
October 2024

Product code:
LXFDB0320

In the magazine
Discover how to make your home brighter and smarter with open source tech, and find out whether the AI PC is anything more than marketing speak. You can also speed up your downloads, tweak network settings, upgrade your graphics card, and emulate both a Psion PDA and the Engima machine. We also have a *Roundup* of the best beginner distros, and so much more to enjoy.

ISSUE 319
September 2024

Product code:
LXFDB0319

In the magazine
Get a flavour of what the latest version of Linux Mint has to offer, and dive inside Linux to discover the audio stack and learn what PipeWire has to do with it. Plus, explore the Linux filesystem, find out about NixOS, expand your storage with RAID, control multiple AI models, and record games. And we haven't even mentioned our packed reviews, Raspberry Pi and news sections...

ISSUE 318
Summer 2024

Product code:
LXFDB0318

In the magazine
Find out how to boost your VPN privacy and set up a free VPN, and continue keeping your secrets safe with our *Roundup* of security-conscious chat services. Plus, try our tutorials on everything from tweaking your install settings to ray tracing on your Steam Deck, and plenty in between. And don't forget to read our packed news, reviews and Raspberry Pi sections to stay up to date with open source.

ISSUE 317
August 2024

Product code:
LXFDB0317

In the magazine
Don't consign your old machines to the scrap heap – they can still be put to work with Linux. And join us as we dive deeper into Linux to explore virtualisation, containerisation and virtual filesystems. We also have tutorials on managing software and files, emulating VDUs, creating old-school pixel art, and more. Plus we have our usual news, views and in-depth reviews.

ISSUE 316
July 2024

Product code:
LXFDB0316

In the magazine
Transform your desktop with hot KDE Plasma, and join us as we start our journey inside Linux to discover how it works. Plus, choose the best text editor for your writing projects, find out about the phone firm making an environmental impact, build your own weather machine, get back to basics with the Linux terminal, keep your video calls private, and enjoy a wealth of reviews, news and views.

ISSUE 315
June 2024

Product code:
LXFDB0315

In the magazine
Discover how to upgrade to Ubuntu 24.04 LTS and join us on a journey through the world's most popular distro's 20-year history. And don't miss our *Roundup* of hacker distros, an in-depth look at the impending Epochalypse, detailed tutorials on live-streaming audio via the terminal, setting up Gnome, editing images in *Krita,* and archiving C64 tapes, and plenty more besides.

To order, visit www.magazinesdirect.com

Select **Single Issues** from the tab menu, then select **Linux Format.**

Or call the back issues hotline on **0330 333 1113**
or **+44 (0)330 333 1113** for overseas orders.

Quote the product code shown above and have your credit or debit card details ready.

MYRIACAT

Credit: https://github.com/myriacat

Radio astronomy – tune in to the aurora

Radio astronomy isn't as difficult as you might think. **Mike Bedford** shows how to get going with just your PC audio system and some free software.

OUR EXPERT

Mike Bedford might never have owned a telescope, but he has always been fascinated by space. So, the opportunity to combine this interest with his passions for electronics and computing was a match made in heaven.

With sunspot numbers set to be high for some time yet, there's every possibility that we'll have other opportunities to see auroras like the ones many people saw in May. However, adverse weather conditions – by which we mean a cloudy sky – would derail any such sightings. So, while seeing an aurora is often a hit-and-miss process, listening to one is less affected by the weather. We're not talking about listening to them with just our ears, though. The northern and southern lights generate radio signals that can be heard with suitable equipment, which might be little more than a PC's internal audio system, an antenna and some suitable software.

It's never going to be an everyday experience unless you live in polar regions, but hearing an aurora might be easier than seeing one because radio waves aren't blocked by clouds.

We're looking at signals at the bottom end of the radio spectrum, which is why they can be received without any special equipment. Wi-Fi signals are transmitted at a frequency measured in gigahertz, as are signals from Bluetooth devices. Mobile phones mostly use gigahertz frequencies, and for conventional broadcast radio (not smart speakers) we find FM in the hundreds of megahertz, going down to the 198kHz of BBC Radio 4's longwave station.

However, while the longwave broadcast band is in the ITU band officially designated Low Frequency (LF), which covers 30kHz-300kHz, the radio spectrum goes much lower. We have Very Low Frequency (VLF, 3kHz-30kHz), then Ultra Low Frequency (ULF, 300Hz-3kHz), Super Low Frequency (SLF, 30Hz-300Hz) and Extremely Low Frequency (ELF, 3Hz-30Hz). Much of this comprises frequencies we consider audible, which is the key to receiving them without a radio receiver.

A PC's microphone input covers frequencies from below 50Hz to the upper limit of human perception, and much further with high-definition soundcards. So, all we need to do to receive these signals is to swap out the microphone – which converts sound to an electrical signal – for an antenna that converts radio waves to an electrical signal. And that antenna might be as simple as just a length of wire.

A first attempt at receiving signals at the bottom end of the radio spectrum couldn't be simpler – we'll start with a few metres of wire as an antenna. To do so, attach wire to a stereo 3.5mm jack plug. These have three contacts, referred to as the tip, ring and sleeve, in order starting from the end that plugs into the socket. Solder the wire to the tip – although, of the three, only the solder tag for the sleeve is the really obvious one. If you're unsure, insert the plug into the mic socket on your PC and try touching the wire on to the solder tags for the other two in turn while listening on the speaker – see later for details – until you hear a signal.

Iron out the details

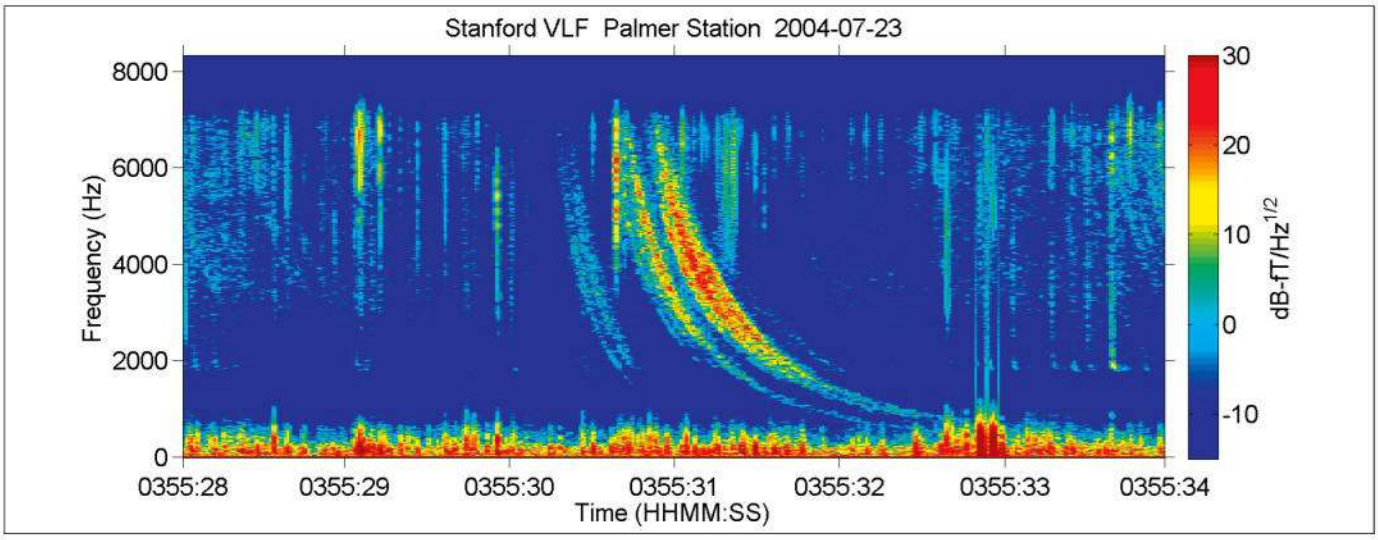
If you don't have a 3.5mm jack plug and a soldering iron, you could use a spare lead that has a 3.5mm jack plug on one end. Cut it in two, and strip back the outer insulation from the part of the lead with the jack plug attached. Now, if present, strip back the woven wire shield to reveal two insulated wires. Alternatively, if there's no shield, you'll find three wires. With either type of lead, strip back the insulation from the two or three wires and attach the antenna wire to the one that connects to the tip. If you don't have a soldering iron, you could use a small connector block. Again, you can identify the correct wire by listening for a signal.

Now insert the plug with the antenna wire attached into your PC's microphone socket and stretch the wire

CREDIT: www.pexels.com/Stein Egil Liland

QUICK TIP

A dedicated VLF/ULF receiver might be more convenient than a laptop. They're easy to build if you have basic electronic construction skills. Take a look, for instance, at the design at https://bit.ly/lxf321radio. This was a NASA project for educational use.



out in your house. Fire up whatever software you use for recording audio – we used *Audacity* – and configure it to listen to the audio from the microphone socket. You should hear a loud buzz. Not a particularly inspiring start, you might think, but you are genuinely listening to a radio signal. In particular, you're hearing the signal created by mains electricity wiring, which will be at 50Hz or 60Hz depending on which country you're in, plus its harmonics – that is, multiples of the 50Hz or 60Hz. While this proves you're receiving a radio signal, these strong mains signals will hinder you from hearing other signals because the harmonics continue up to several kilohertz and can drown out weaker signals.

We need to improve the antenna. One option is to use a longer length of wire. However, unless you take adequate precautions, you'll also receive a strong mains signal, so you'd be back at square one. We'll consider some possible solutions in the next section.

A better setup

First, you could take the antenna wire out of the house to get it further from mains wiring. If you do that, you should use coaxial cable (often abbreviated to coax) from the jack plug to the start of the external antenna. Since coax is shielded, it won't pick up signals, so only the external wire acts as the antenna. The coax should start at the jack plug, not at the end of an unshielded lead, the shield of the coax should connect to the sleeve, and the inner conductor of the coax to the tip. At the far end, the wire antenna should be connected to the inner conductor of the coax.

You might want to read up on coaxial cables. A long outdoor antenna could, in some circumstances, pick up a large enough voltage to damage a PC's audio circuitry. One way to prevent that is to wire a couple of diodes (any type), connected opposite ways round, between the tip and the sleeve of the jack plug. For extra protection, use a cheap USB soundcard instead of your PC's audio circuitry, but still use those diodes.

Next up, even though moving the antenna outside will certainly reduce mains pickup, if you're in a built-up area, these signals could remain problematic. The obvious solution, therefore, is to use a laptop and take it into a remote area. Serious VLF enthusiasts have taken this solution to the extreme, setting up their stations in the deserts of Arizona. However, even if you do go to a rural area, it would still be wise to separate the antenna from your laptop using coax, because PCs also generate potentially interfering VLF signals.

A long piece of wire is OK to start with, but if you get serious, you really ought to consider a better antenna. Generally speaking, long wire antennas are good for low frequencies, which implies long wavelengths. This is because the efficiency of an antenna depends on how long it is as a percentage of the wavelength. But once we get to even the top end of the VLF band, the wavelength is 10km, so any practical antenna is hugely inefficient. And it gets worse. This rises to 100km for the bottom end of the VLF band and 1,000km at the bottom of the ULF band, which is probably as low as you'd want to go. So, VLF/ULF enthusiasts tend to abandon wire antennas – however long they might be – and opt instead for a whip or loop antenna, together with some external circuitry to boost signals before they reach the PC's audio circuitry. This is beyond the

Whistlers are impressive-sounding sferics. Like most spectrograms, this one represents time horizontally, so it will look different on a WebSDR or in Myriacat.

» NATURAL SIGNALS

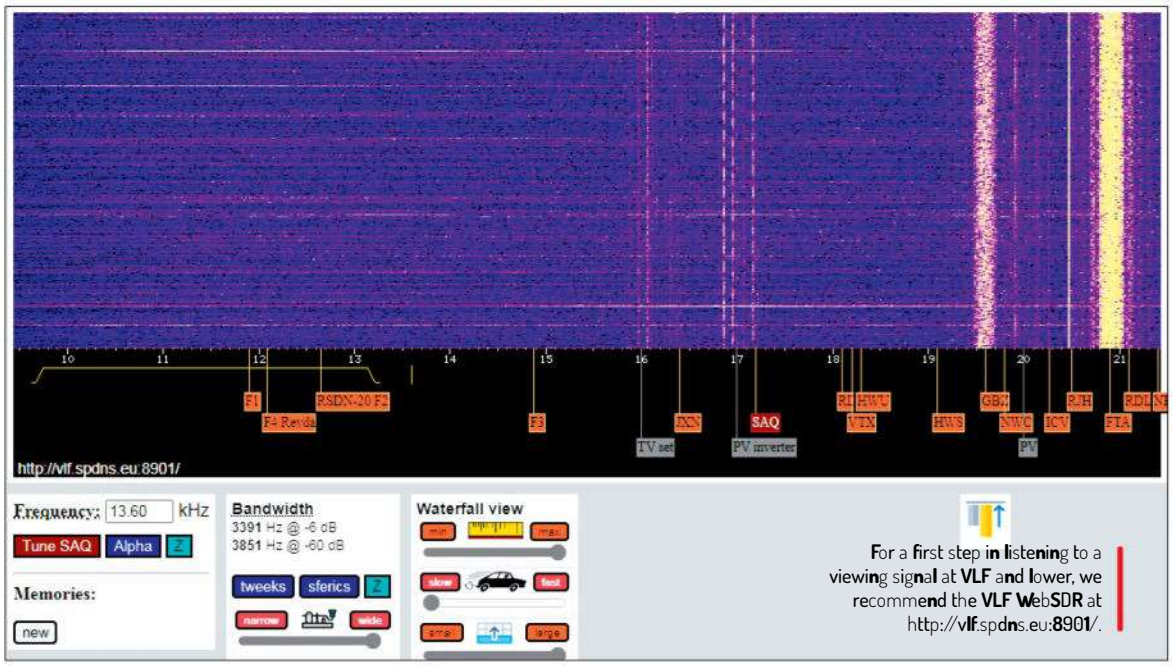
Natural VLF signals are commonly referred to as sferics, short for atmospherics, because they originate in the atmosphere. To start, we'll take a look at those sferics caused by lightning.

In addition to light, lightning produces broadband radio signals. However, those at VLF/ULF are the most interesting and they take various forms. Confusingly, the most common form of sferic tends to be called a sferic. These are found at frequencies up to 10kHz or so, they are caused by lightning strikes within a thousand kilometres or so, and they are commonly described as sounding like frying bacon or cracking twigs. They are horizontal lines on waterfall displays. Next up are tweeks, which are caused by more distant lightning and propagate through the ionosphere. Because the higher frequencies arrive before the lower frequencies, the tone decreases in frequency from 10kHz to 2kHz in less than a second, sounding like a ricochet. Then there are whistlers, which take a longer path than tweeks, travelling through the magnetosphere, and sound like a tone decreasing in frequency over a second or two. Whistlers are more common in winter, and ideally you should listen around dawn.

And finally we come to the auroral chorus. This originates in the magnetosphere and has been likened to the chirps of crickets or frogs, or the calls of tropical birds, whales or dolphins. Listen between 600Hz and 1.6kHz.

QUICK TIP

As an alternative to our featured *Myriacat* software, you might like to delve into *VLF Receiver Software Toolkit* (http://abelian.org/vlfrx-tools/). It's a modular software toolkit for time-stamped signal processing.



scope of this article, but you'll find lots of information, including practical designs, on the web.

Next, we need to say a few words on the PC's audio circuitry or, perhaps, an external soundcard. Other things differentiate a good soundcard from a mediocre one, and you might like to consider these if you get serious, but an important consideration is the sampling rate. It isn't an issue if you just want to receive signals relating to natural atmospheric phenomena, but it is if you want to delve into some of the man-made signals in the higher part of the VLF band and into the LF band. The highest frequency signal that can be converted by an ADC is half its sampling frequency. So, 44.1kHz audio circuitry can handle signals up to just over 22kHz. However, 96kHz circuitry takes you up to 48kHz, and 192kHz circuitry doubles this again.

We'll look later at suitable software for processing the received signals, but to give you a better idea of what you need from software, let's look at WebSDRs, which offer an easy way to receive VLF/ULF signals

QUICK TIP

A rather quirky VLF station is SAQ in Sweden. This radio station uses the only remaining Alexanderson transmitter that predated electronic transmitters. Since the generator has 488 poles, it produces an AC signal on SAQ's 17.2kHz transmitting frequency. It transmits Morse code occasionally – see https://alexander.n.se/en/.

The online option

If you don't fancy the idea of setting up your own VLF/ULF radio station, there's an alternative option. In fact, even if you are intent on building your own receiving setup, this option will help you to hear and hence recognise the types of signals you might encounter. It will also show you the features you'll need from software when you setup your own station.

Online you can find lots of WebSDR (software defined radio) receivers that radio amateurs have set up for others to use. You can find a list of them around the globe, together with links, at **www.websdr.org**. Not all cover the VLF band and below, but quite a few do, so it's these you should head over to. The one at Twente University in the Netherlands is one such SDR, and is mentioned often by VLF enthusiasts. Another particularly interesting one, in being dedicated to VLF and below, is at **http://vlf.spdns.eu:8901/**. It appears to make a very good job of eliminating mains interference, and it even has buttons labelled Sferics and Whistlers that provide a shortcut to the frequency and bandwidth of these two atmospheric phenomena. You'll nearly always be able to hear sferics, although we did find that the SDR wasn't always available.

WebSDRs present signals both audibly and visually. The visual element is a so-called waterfall display. The vertical axis represents time, with the most recent time at the bottom, while the horizontal axis represents frequency, and the amplitude of signals is represented by colour. Depending on how much you zoom in on the horizontal axis, you can see several signals on different frequencies. To listen to one, click on it in the waterfall display. You can also select the bandwidth – how large a range of frequencies to listen to – and you choose this based on the type of signal you're listening to.

Software processing

Previously we suggested using *Audacity* or similar as a first step, as it provides an easy way of demonstrating that you can receive VLF or ULF signals and, in particular, those caused by mains electricity supplies. However, although it can display a waterfall display and do some filtering, its tools are mostly intended for working with audio tracks you've already recorded. What we really need is a means of handing signals in real time, in much the same way as with WebSDRs.

We were surprised to discover that Linux software that can do that is thin on the ground. The one we eventually found, though, does a good job. It's called *Myriacat*, you can get it from **https://github.com/myriacat** and it's totally free for any purpose, even though it's not open source. There's nothing to install, just download the .tar.gz archive, extract the contents and run the executable. The main downside is that there's no user documentation so, to a degree, it involves trial and error. The main things you need to master, though, are changing the frequency you're listening to (the red line), the bandwidth (the semi-transparent grey area next to the red line), and the sensitivity – the degree to which signals are amplified. Despite the lack of a user manual, a limited help screen

appears when you first run *Myriacat,* and you can display this later by pressing `i`. There's also useful information under the heading of FAQ on the GitHub. We struggled to reverse some of the settings we made, but if this happens, just delete the *Myriacat* executable and regenerate it from the .tar.gz archive.

If you fancy delving deeper, other software you could try – although neither offers a turnkey solution – is *VLF Receiver Software Toolkit (see Quick Tip on page 66),* and *GNU Radio* (**www.gnuradio.org**). With *GNU Radio,* the sky's the limit, but building your system – a process that involves wiring together on-screen functional blocks – isn't always a trivial process.

So, you're probably wondering what we managed to see or hear. First of all, though, let's describe the setup. It was a modest station that pretty much anyone could put together. The receiver was the standard audio circuitry in an ordinary PC with a maximum sample rate of 44.1kHz. The antenna was about 15m of wire, at a height of about three metres. However, it was separated from the nearest building by about five metres and connected via coax. Compared to the original setup of an indoor antenna, this was a huge improvement in reducing the interference from the mains electricity supply. Since we didn't use a specially designed VLF/ULF antenna nor a pre-amplifier, though, and we didn't adjourn to some remote area, there would be plenty of scope for further improvement.

First, we were able to see and hear regular sferics, frying bacon sounds and all. These could be seen in *Myriacat* as horizontal lines. We do have to admit, though, that we didn't hear an auroral chorus, but that's because there were no auroras in the UK when our station was operational. Nor did we hear the more esoteric types of sferic, namely tweeks and whistlers. The main reason for this is that we wanted to give you as much of the prime winter season as possible to set up your receiving station and listen to the broadest range of sferics, which meant this article had to be written in a sub-optimal time of the year.

We were also able to see several man-made signals, which appeared as vertical lines of various widths. We saw signals at the following frequencies that appear against a tentative identification (some more tentative than others) comprising a callsign and location, plus a distance from our setup in northern England. The identifications were drawn from online sources, and the frequencies quoted are the official ones from those lists rather than the approximation we'd have obtained from *Myriacat.* However, lists differ, some are quite old, and particular callsigns occasionally move location. What's more, most of these stations are military installations, so official information is often in short supply – hence the uncertainty.

The strongest signal was almost definitely GQD at Anthorn in Cumbria, on 19.6kHz, which is used for submarine communication, and was about 145km away. Others, in order of decreasing confidence in the identification, were as follows: 21.75kHz, HWU, Rosnay, France, at a distance of 580km; 20.9kHz, FTA, Seine-Port, France, 639km; and 18.1kHz, RDL, Krasnodar, Russia, 3,064km. We also noticed several weak signals between 13kHz and 14kHz, and while it's possible that they were interference caused by electronic equipment, we note that several Russian navigational stations operate in this frequency range.

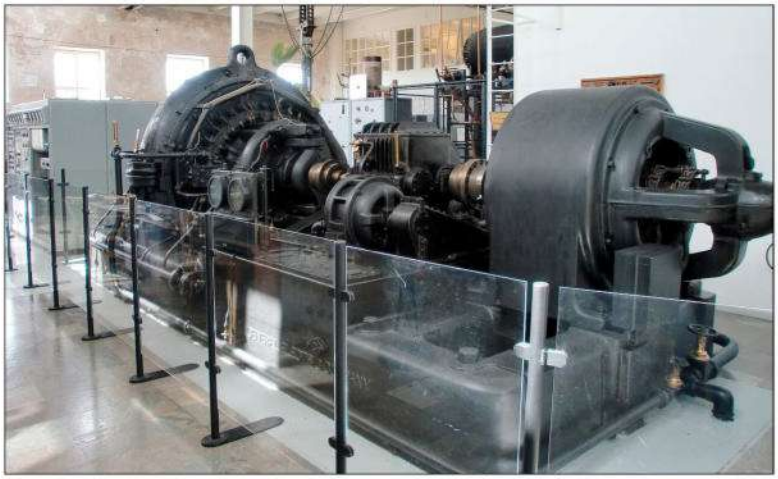
Although VLF signals are able to propagate very long distances, with a simple setup you're most likely to find man-made signals up to about 3,000km from home. So, if you live outside Europe, you'll probably encounter different man-made signals from the ones listed above, and that gives you the opportunity to identify them with a bit of detective work. And exactly the same applies to our main theme of listening to nature's radio signals. There is no one size fits all, so do experiment, research and try out alternative solutions, and you might just find yourself listening to an aurora while watching its light show in the sky. LXF

» MAN-MADE SIGNALS

Natural signals aren't the only ones you'll find at the bottom of the radio spectrum – there are also man-made signals. You'll just hear bleeps, but these signals transmit at VLF because of the unique characteristics of this region of the radio spectrum.

Long-range transmission is possible at VLF but, unlike shortwave, the propagation characteristics are more stable and predictable. So, several stations in the VLF and lower part of the LF band transmit accurate time information. Several have closed down because GPS can now provide this service, but some are still operational for use in so-called GPS-denied environments. If you have a soundcard with a 96kHz sampling rate, there are quite a few you could receive, but it appears that the last one to transmit on a frequency low enough for a 44.1kHz soundcard to receive closed down several years ago.

Another aspect of VLF signals that doesn't apply to higher frequencies is that they can penetrate a few tens of metres of seawater. For this reason, submarine communication is conducted at VLF. Most stations are beyond the frequency range of a 44.1kHz soundcard, although a few Russian navigation stations operate between about 12kHz and 15kHz, and GQD in Cumbria, UK, is on 19.6kHz. Lists of VLF stations are available online to help identify any stations you encounter, but do bear in mind our comments near the end of the article about the accuracy of these lists.



This electro-mechanical contraption is the transmitter of the Swedish SAQ station, which still transmits, occasionally, on 17.2kHz.

CREDIT: Dagjoh, CC BY 3.0, https://commons.wikimedia.org/wiki/File:Alexandersson

DISPLAYS

Upgrade it: Enhance your viewing pleasure!

It's worth spending more on a good display, because you could be using it longer than any other component, says **Neil Mohr**.

OUR EXPERT

Neil Mohr spent so long tweaking Amiga displays back in the day that he's almost blind now – or is that genetics?

You'll spend countless hours staring at it and you'll likely still be using it long after you consign your GPU or CPU to the ewaste bin. We're talking about your computer's visual display unit. The humble monitor has received a dazzling number of upgrades over the last decade or so – after the awkward long shift to flatpanels and 1080p resolutions, the display world was suddenly inundated with 4K UHD, HDR, 144Hz refresh, OLED, MicroLED, curved screens, ultra-wide screens, USB hubs, dynamic sync and more.

Now that's a wide screen!

As with most things, choosing a display is a combination of what you want to use it for and your budget. We've got a selection that costs from £110 up to almost £4,000, but most people won't need an 8K display. Let's start by looking at panel types and going from there.

First, determine your monitor's main purpose: gaming, professional or general use. Generally, gamers should prioritise fast refresh rates and low response times; professionals should prioritise colour accuracy; while general users have less specific needs but often opt for a monitor with a high-contrast panel.

Resolution and size are important. More pixels mean a better quality image but require more GPU power to push them around; old systems can struggle with 4K. Bigger panels are easier to read but harder to fit on a small desk. Remember, the larger the screen the lower the pixel density, so a 1080p resolution on a 32-inch display may look chunky but is OK on a 24-inch screen.

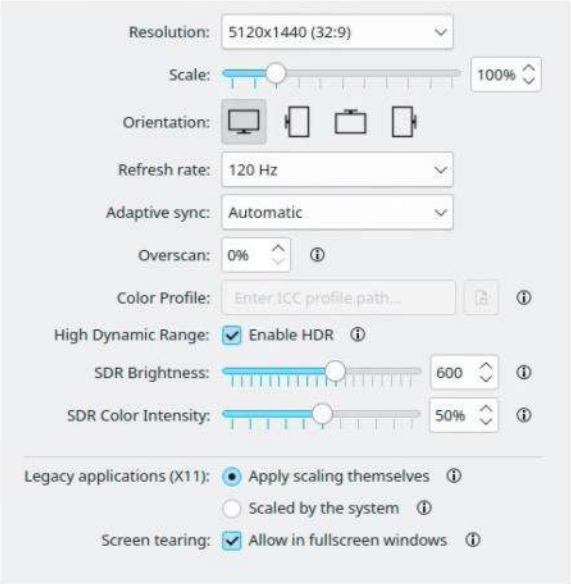
Aspect ratio defines how wide a screen is compared to its height. Usually the standard is 16:9 for the default 1920x1080 resolution. However, there was a trend to offer wider ratio displays, sometimes called ultra-wide, pushing this to the old cinematic 21:9. Recent panel innovations enabled a few models to take this to the extreme of 32:9 and the odd 5120x1440 resolution.

» CHAINED TO THE SYNC

We've had vertical sync, aka vsync, for years. It ensures your screen is only updated during the vertical blank period, when the display finishes drawing at the bottom of the screen and is returning to the top. Enter gamers, who demand the fastest screen updates – vsync is turned off to achieve this, and suddenly people see on-screen 'tearing', which is the screen being updated mid-refresh. The problem is exacerbated with high-refresh-rate displays – 120Hz and higher.

To avoid this, AMD came up with FreeSync, while Nvidia has G-Sync. They are hardware/software solutions that need to be supported in the graphics hardware, monitor, OS (the driver) and game for it all to work. They ensure the screen is only updated when a full frame has been drawn but it's done as quickly as possible. They retain support for super-fast screens up to 360Hz for G-Sync and 200Hz for FreeSync Premium, or 144Hz for older FreeSync.

Both systems require a DisplayPort connection to work on top of a compatible monitor. Most distros (including Ubuntu 16.04+) directly support AMD FreeSync and should default to On with kernel 6.5 onwards. The Nvidia Linux driver supports G-Sync but you must switch it on via its desktop utility with Display Configuration > Advanced > Allow G-Sync. Also, under OpenGL, ensure Allow Flipping and Allow G-Sync Compatible are active.



The latest KDE 6 display settings with HDR options on offer.

If gaming is important, maximum refresh rate should be a key feature. The base speed is 60Hz, which is fine for general desktop use but even here 120Hz makes for a smoother mouse cursor. Response time tells you how long a monitor takes to change individual pixels from black to white (though GtG response time is from one shade of grey to another). Longer response times can mean motion blur when gaming or watching fast-paced videos. For most monitors, the highest response time you'll likely see is 5ms, while the fastest gaming monitors can have a 0.5ms response time.

Until recently, all panel technology was LCD-based and you'd choose from TN, VA or IPS. We recommend avoiding TN panels; they can refresh the fastest, but usually a lower display quality doesn't warrant it. Generally, IPS offers the best colour reproduction but lower refresh speeds. For general or gaming use, VA offers the best contrast and is faster than IPS.

More recently, OLED and MicroLED displays have appeared. For many, OLED is the de facto panel to have, due to the response and amazing colour, but they're incredibly expensive. While MicroLED options are very rare, they are still really only available for TVs.

What about HDR? Windows users have been able to use this for a while, it has (in part) made it the Linux desktop – but only for KDE 6 Wayland sessions. So, if you're not currently running a distro that supports Wayland, HDR isn't a priority, though it's getting tentatively closer. And using Steam's *Gamescope* software, you can even run HDR games on Linux.

Professional users have special needs. If you're a photographer, print proofer, web designer, special effects artist, game designer or someone who needs precise colour control, this part's for you...

Monitors vendor-certified as colour accurate cost more but are worth it. If you want an accurate display out of the box, this is your best choice, especially for monitors without calibration capabilities. Pro monitors should come ready for work with no adjustment required. A DeltaE (dE) value of 2 or lower is good – under 3 is considered invisible to the human eye.

You want calibration options. There are two ways to accomplish this: the on-screen display (OSD) and software. Calibration options should include choices for different colour gamuts, colour temperatures and gamma curves. At minimum there should be sRGB and Adobe RGB standards, colour temperatures ranging from 5,000 to 7,500K, and gamma presets from 1.8 to 2.4. Monitors used for TV or movie production should also support the BT.1886 gamma standard.

A final point, often overlooked, is the monitor stand. Lower-priced models only provide a fixed stand with no tilt or height adjustment, unless you plonk it on top of *War and Peace*. More expensive models offer stands with reasonable adjustments and some can even adjust to a portrait mode. However, they're still sitting on your desk. Pretty much all displays use the VESA mount, though, which means you can invest in a suitable clamp mount and have your monitor 'float' over the desk, or wall-mount it. As the VESA mount is universal, it shouldn't matter if you upgrade the display down the line, as you'll be able to reuse the stand.

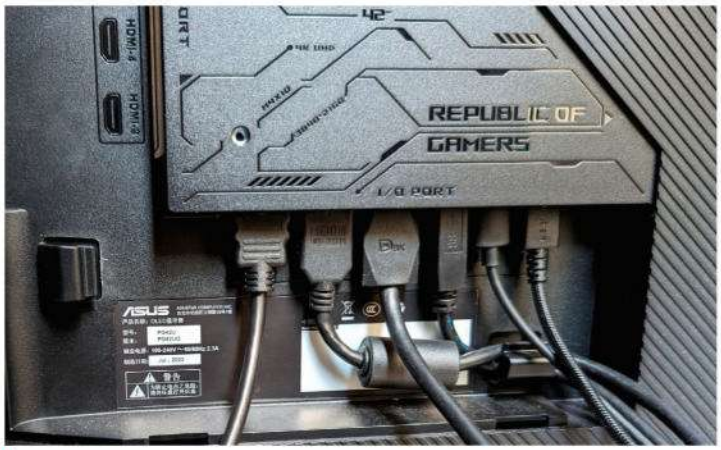
ANY PORT IN A STORM

Do we need to tell you the history of computer video connectors? The short version would be something like: 15-pin D-sub VGA > DVI > HDMI > DisplayPort, and we should probably mention USB-C... The issue with VGA was it was analogue-only, so had to go. The issue for DVI is that it retained backwards compatibility for VGA analogue lines, while ultimately it became bandwidth limited, and couldn't support high-refresh high-res displays – it was envisaged for 1080p @ 60Hz, but a dual-link mode did bump it to 2560x1600.

HDMI was introduced as a digital-only consumer connector, so it also carries audio (DisplayPort does, too) and Ethernet lines, while supporting more display modes and, requires a substantial licence payment per port and support for DRM HDCP. As of 2023, the latest version of HDMI, 2.1b, offers a maximum bandwidth of 48Gbits/s supporting up to 10K at 60Hz with compression.

DisplayPort implements a new packet-based transmission protocol, though compatibility with VGA, DVI, HDMI and USB-C is available via adaptors. The latest version, as of 2024, is 2.1a, supporting 80Gbit/s bandwidth and 8K up to 85Hz with HDR.

USB-C can support displays, known as DisplayPort over USB-C – it sends the DisplayPort data packets just over the USB-C interface. It seems the industry is moving towards using USB-C on laptops for external display connections. Be careful buying USB-C-to-HDMI hubs; some only support 30Hz at 4K, and we've found some cutout – we had better luck with a direct USB-C-to-DisplayPort adaptor cable.



HDMI, DisplayPort, USB-C – there's probably a power connector in there, too...

Pixio PXC277 Advanced

The best budget 1440p display to grab right now.

SPECS

Size: 27-inch
Resolution: 2560x1440
Type: VA
Rate: 165Hz
Sync: FreeSync, G-Sync
Brightness: 320cd/m²
Speed: 1ms
Contrast: 4000:1
Angle: 178/178
HDR: Yes
Colour: 124% sRGB, 97% DCI-P3
Ratio: 16:9

The reason we've picked the Pixio PXC277 Advanced as the best budget 1440p monitor is because it's such a solid all-rounder for the price. No unnecessary bells and whistles, just a monitor that delivers in all the right areas.

For a pretty cheap price tag, you get a 27-inch, 165Hz monitor with a claimed 1ms response time and up to 320 nits brightness. This brightness is higher than many more expensive VA panels.

Naturally, that 1ms response is a best-case scenario, and in practice, as we'd expect from VA, it's not quite so snappy. But if you get the settings right (overdrive set to low), there's little overshoot or ghosting, and it feels quick.

In practice, all this makes for a monitor that should suit most people even gamers. It's sufficiently fast and snappy, and, most importantly, you're getting a damn good picture straight out of the box. Although it has a predictably lacklustre HDR experience, the monitor's pretty accurately calibrated and things look reasonably vibrant for such a cheap monitor, with the lovely deep blacks and stellar contrast that you'd expect from a VA panel.

To top it all off, this thing doesn't look budget at all on the desk, thanks to its slim bezels and tri-leg metal stand. We reckon this would be a great addition to anyone's desk, and certainly for the price. ***Dave James***

A fine-looking curved display that'll tick most people's boxes.

VERDICT

DEVELOPER: Pixio **PRICE:** £200

» Rating 9/10

Gigabyte M32UC

An "entry-level" 4K high-refresh gaming display.

SPECS

Size: 31.5-inch (curved)
Resolution: 3840x2160
Type: VA
Rate: 144Hz (160Hz OC)
Brightness: 350cd/m²
Speed: 1ms
Contrast: 3000:1
Angle: 178/178
HDR: VESA HDR400
Colour: 123% sRGB, 93% DCI-P3
Ratio: 16:9

Multiplatform gamers looking for a high-performance 32-inch gaming monitor for 4K at 144Hz gaming will find a lot to appreciate about the M32UC from Gigabyte. With a respectable number of ports and other useful features, along with snappy pixel response time and great colour gamut coverage, this is a great-looking monitor with satisfying performance.

The only thing about it is that out of the box, we found it presenting a fairly flat and dimly lit display image, even when HDR is enabled. Calibration is certainly necessary. We had to fiddle with its brightness, contrast settings, colour controls and HDR settings a little bit, but when we did, the visual experience it offered was incredible.

There are two HDMI 2.1 ports and one Display Port 1.4, alongside three USB-3.2 downstream ports and one 3.2 upstream port. Rounding out the ports is an 3.5mm earphone jack, which is definitely needed because the external speakers are very poor. Beyond issues with overall volume, underwhelming mid-tones and cracking bass add to general uselessness. It's definitely undeniable where cuts were made.

The HDR400 support is all right, although it's nothing special, but when you factor in the excellent price for this kind of curved display, you are getting one of the best 4K monitors for gaming by value on the market at the moment. Just don't expect much from the built-in speakers here, because they barely get the job done. ***John Loeffler***

We think this monitor might be aimed at gamers.

VERDICT

DEVELOPER: Gigabyte **PRICE:** £500

» Rating 9/10

CREDIT: Gigabyte, Asus

Asus TUF Gaming VG289Q

The best choice for budget 4K gaming.

SPECS

Size: 28-inch
Resolution: 3840x2160
Type: IPS
Rate: 60Hz,
Sync: FreeSync
Brightness: 350cd/m²
Speed: 5ms
Contrast: 1000:1
Angle: 178/178
HDR: VESA HDR400
Colour: 90% DCi-P3
Ratio: 16:9

The Asus TUF Gaming VG289Q offers tremendous value for gamers looking to take the step up to 4K resolution without having to go completely bonkers on price. The VG289Q has a competent overdrive function, which helps to reduce motion blur, and we didn't notice any ghosting on the panel.

AMD FreeSync is supported here, although you should note that the maximum refresh rate is just 60Hz (FreeSync will work down to 48Hz on this monitor) and offers a single DisplayPort 1.2 and twin HDMI v2.0 ports.

We can happily say that SDR content looks rich and colourful on the VG289Q, but shifting to HDR doesn't really show any improvement in image quality. It does come with a decent stand, which can be unusual these days, and this offers height adjustment alongside horizontal or vertical standing, with pivot, tilt and swivel adjustment.

If you're just looking to dip your toes in the 4K gaming monitor market, it's hard to find any serious faults with the VG289Q. ***Michelle Rae Uy***

Some might say this is no frills, we just think it delivers solid 4K performance.

VERDICT

DEVELOPER: Asus **PRICE:** £279

» **Rating 8/10**

BenQ 23.8" MOBIUZ EX240N

1080p displays aren't dead or buried yet!

SPECS

Size: 23.8-inch
Resolution: 1920x1080
Type: VA
Rate: 165Hz
Sync: FreeSync
Brightness: 250cd/m²
Speed: 1ms, 4ms GtG
Contrast: 3000:1
Angle: 178/178
HDR: HDR10
Colour: 24-bit, 72% NTSC
Ratio: 16:9

This is the budget 'N' model of the EX240 range. It uses a cheaper VA panel versus the IPS of its EX240 brother. It also saves you around £90. Price aside, you get the same 165Hz refresh rate, video inputs (a single DisplayPort 1.2 and dual HDMI 2.0 ports) and HDR10 support.

Thanks to the inherent contrast advantage of VA panels over IPS, you could argue that basic HDR10 support is a better fit here and likely to deliver something closer to an actual HDR experience. On the other hand, BenQ only rates this VA model at 250 nits, which is low by today's standards and hardly bodes well for HDR sizzle. You also don't get a USB hub of any kind, while the non-N has a dual-port USB-A hub.

On paper, however, arguably the biggest worry is pixel response. BenQ rates the MOBIUZ EX240N at 4ms for grey-to-grey response. That's some distance off the 1ms of the best IPS panels. Indeed, response is typically VA technology's greatest weakness.

BenQ does offer tweakable pixel overdrive; though, oddly, on this panel it's only available when running the screen in the distinctly oversaturated Game mode.

The EX240N does offer reasonable response. It's not a mess by any means . The fact that you have to run it in Game mode and suffer the colour issues undercuts that faint praise. But a 165Hz gaming monitor for £110 from a good brand like BenQ seems like one heck of a deal. ***Jeremy Laird***

A balanced panel that delivers speed without the price.

VERDICT

DEVELOPER: BenQ **PRICE:** £110

» **Rating 7/10**

Dell S3221QS

Best all-round choice if you fancy curves in your life.

SPECS

Size: 31.5-inch (curved)
Type: VA
Resolution: 3840x2160
Rate: 60Hz
Brightness: 300cd/m²
Speed: 4ms (GtG)
Contrast: 3000:1
Angle: 178/178
HDR: Yes
Colour: 99% sRGB, 90% DCI-P3
Ratio: 16:9

The Dell S3221QS is a gorgeous monitor inside and out. Its simple yet elegant silver design gives it a unique look that sets it apart from all the gamer-centric or boring black office monitors you're more likely to find. It comes with a stunning 4K VA panel to match, which looks great and delivers colours that we found to be accurate and rich, as well as crisp, detailed image quality. Because of the 1800R curvature, even after hours of use, there's no eye strain.

And, to make it an even better proposition, it's got some great features as well – namely, decent-sounding speakers and an interesting picture-in-picture functionality that enables you to display two different computers in the same display. That's pretty nifty and a great way to make up for the fact that there's no USB-C connectivity. This model is a couple of years old at this point, but the past two years have been fairly slow for monitor developments, especially on the more mainstream and business-user side of things, so it can hold its own against the best business monitors out there.

It does offer an integrated USB hub that provides USB 3.0 upstream and downstream ports alongside BC 1.2 charging support. For video input, there's two HDMI and a single DisplayPort 1.2 input. ***John Loeffler***

A nicely balanced 4K big-panel display.

VERDICT

DEVELOPER: Dell **PRICE:** £359

» Rating 9/10

BenQ SW321C PhotoVue

A 4K display targeting photo and video professionals.

SPECS

Size: 32-inch
Resolution: 3840x2160
Type: IPS
Rate: 60Hz
Brightness: 250cd/m²
Speed: 5ms (GtG)
Contrast: 1000:1
Angle: 178/178
HDR: HDR10, HLG
Colour: 100% AdobeRGB, 100% sRGB, 95% P3
Ratio: 16:9

Pro-level displays are no longer the premium priced, inaccessible purchase they once were. At least as far as the BenQ SW321C PhotoVue is concerned. This 32-inch 4K photo monitor is a step or two up in terms of both performance and usability, featuring an incredibly wide colour gamut of 99% of the Adobe RGB colour space and 95% of DCI-P3.

If you're in the cinematography or photography sphere, that's exactly what you need. To test this monitor's performance, we ran software-based calibration – and its scores for colour gamut, tone response, white point uniformity, contrast, luminance uniformity, colour uniformity and colour accuracy were all excellent.

The main connection bay is located along the underside of the monitor and includes dual HDMI 2.0 ports, USB Type C and Type B, and DisplayPort. This arrangement makes for tidier cable runs – there's even an aperture in the lower section of the stand to keep everything together – but access isn't quite so easy. Somewhat more accessible is the secondary set of ports inset on the side of the monitor – two USB 3.1 and an SD card reader.

The BenQ SW321C PhotoVue monitor might be a little bit older than most, but it's still got serious performance chops for professional users, and the fact that it's just slightly out of date means that you're likely to find it on sale at a reduced price at a lot of retailers. And that's on top of all the other features this monitor boasts. ***John Loeffler***

It's going to cost you but you won't get a better professional-class display.

VERDICT

DEVELOPER: BenQ **PRICE:** £1,800

» Rating 9/10

CREDIT: BenQ, Dell, Asus

Asus ROG Swift PG27AQDP

The best gaming OLED display – but at a price.

SPECS

Size: 26.5-inch
Resolution: 2560x1440
Type: OLED
Rate: 480Hz
Sync: FreeSync, G-Sync
Brightness: 1,300cd/m²
Speed: 0.03ms (GtG)
Contrast: 1.5m:1
Angle: 178/178
HDR: HDR10
Colour: 99% DCI-P3
Ratio: 16:9

The Alienware AW2725DF has been one of our favourite high-performance OLED monitors thanks to its excellent image quality and fast response. However, there's a new king of the castle: Asus's ROG Swift PG27AQDP. It takes the basics of the AW2725DF and adds even more performance.

The ROG Swift PG27AQDP features the same 2560x1440 resolution but boosts the maximum refresh rate from 360Hz to 480Hz while maintaining Nvidia G-Sync and AMD FreeSync compatibility. We were impressed with the colour calibration straight from the box (no further calibration was necessary), and we saw a maximum SDR brightness of 400 nits. While that is not as high as you'll see in many IPS panels with MiniLED backlighting, it's very bright for an OLED panel. In addition, HDR brightness peaked at 1,300 nits. As expected for an OLED, black levels and dynamic are unmatched.

All of this performance comes at a price, however, with the ROG Swift PG27AQDP ringing in at £949, while the competing Alienware AW2725DF will set you back only £780. But if you have the hardware to push the ROG Swift PG27AQDP to its fullest potential, it's one of the best options available for those looking at high-performance OLED panels. ***Brandon Hill***

If you're serious about high frame rates, start saving up!

VERDICT

DEVELOPER: Asus **PRICE:** £949

» Rating 10/10

Dell UltraSharp UP3218K

For extreme 8K resolution and price, this is for you!

SPECS

Size: 32-inch
Resolution: 7680x4320
Type: IPS
Rate: 60Hz
Brightness: 400cd/m²
Speed: 6ms (GtG)
Contrast: 1300:1
Angle: 178/178
HDR: No
Colour: 100% AdobeRGB, 100% sRGB, 100% Rec 709, 98% DCI-P
Ratio: 16:9

We don't usually run into technology that's so far ahead of the curve that we're left dumbfounded, which is why the Dell UltraSharp UP3218K has impressed us even more. Finding one of the best monitors that can reach the raw gorgeousness this one can should be next to impossible. It's not just the resolution, either.

Dell went so far as to ensure that the build quality and colour reproduction are the best in the business as well. The monitor is factory calibrated, so it looks great without any tinkering, and that ultra-high resolution, along with the colour handling of the monitor, makes any content look incredible – so incredible that sometimes footage almost looks 3D with the amount of detail you're getting. It offers two DisplayPort inputs and has an integrated USB 3.0 hub with an upstream connector and three USB downstream ports, one of which doubles as a BC1.2 charging port.

The Dell UltraSharp UP3218K is aimed at professionals, obviously, so if that sounds like it's made for you, it's probably the best monitor you'll ever find. This product is only available in the US and UK right now, and given that it's a couple of years old at this point, it's harder to find than most. ***John Loeffler***

No one said 8K displays would be affordable...

VERDICT

DEVELOPER: Dell **PRICE:** £3,200

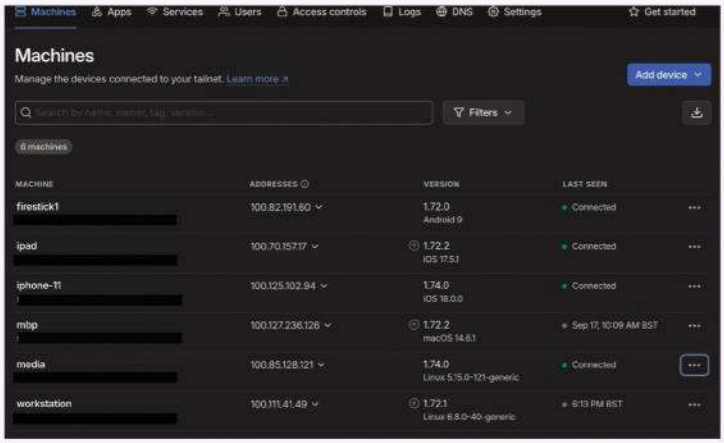
» Rating 8/10

Simplify network configs with Tailscale

No one calls **Stuart Burns** a simple man, but he appreciates tools that simplify his admin life, especially for networking.

Using Tailscale daily has been quite a revolution – so, to share the goodness of Tailscale with *Linux Format* readers, we're going to walk through a basic working setup, though there are many more advanced configurations. We are creating a working Tailscale network environment with our devices connected to the network. There are two significant steps to setting up Tailscale.

First, create a Tailscale account using the Tailscale website at **https://bit.ly/47B2ADZ**. You also need an account with one of the federated providers (Microsoft, Google, GitHub and so on), as Tailscale doesn't want to know your details (as a security choice it made).



This is the console that forms the backbone of the Tailscale management interfaces. APIs can be used, too.

It asks a few basic questions. By default, you are assigned a Tailscale network, aka a tailnet, in the form of (**xxxx.ts.net**). This is your own network, separate from everyone else – the logical boundaries if you like. That's the hard bit done.

Pin the Tailscale

Once the account is open, you can download and install the clients. Alternatively, visit **https://bit.ly/3ZshZo8** for the client downloads.

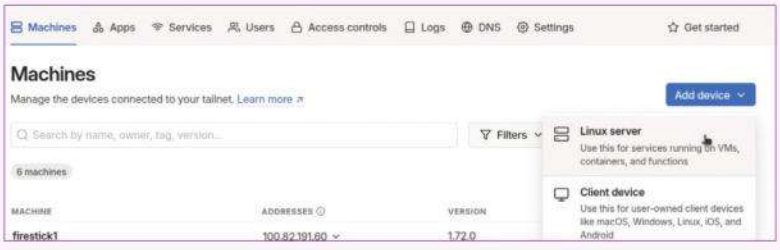
The Linux client, somewhat unfortunately, comes only as a shell script install (`curl -fsSL https://tailscale.com/install.sh | sh`)

During installation, the installer takes the current hostname of the device in question and uses that hostname as the hostname on the Tailscale network. Start up the Tailscale client by using:

```
$ sudo tailscale up
```

On completion, it requires you to log in and approve adding that client to the tailnet. Click to accept and it completes. On Linux desktops, there will be a URL on the screen. Copy this link and paste it in. It requires you to go through the authentication process again.

Depending on the scenario, there are different ways in which Tailscale can be installed.



To install Tailscale on additional devices, log in to the Tailscale website to download the appropriate client and install it as you would any other application. At this point, to test the tailnet, use the *ping* command to test whether it is up:

```
$ ping <tailnet member>
```

If it is working, the address it replies with is a non-local address.

Of course, installing it on Mac OS and Windows is easy and takes the standard application installation (but does require admin rights). A new GUI toolbar icon is created that enables you to configure items.

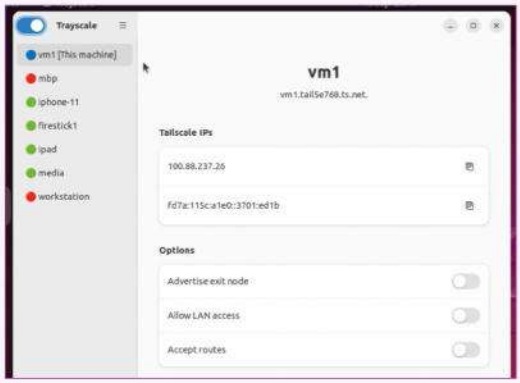
Sadly, for Linux computers, the installation has no official GUI, but once downloaded and installed, it does auto-start. Fortunately, a Linux developer has created a functional GUI client for the Linux desktop called *Trayscale* (**https://bit.ly/3XsKGid**). It is a Flatpak application *(see boxout, right)*, but it is worth it. Although being basic, it provides most of the options available in the standard clients.

Once there are several Tailnet clients installed and running, you should be able to ping the other clients using their hostnames and connect to services as though the machines in question were all local. Amazing right? That is all thanks to MagicDNS – see **https://tailscale.com/kb/1081/magicdns**.

The great thing is that there's a huge array of supported devices, including Amazon Firesticks, Raspberry Pis, and certain TV brands.

So, putting it all together, all the devices to which you add the client become members of the network. The IP addresses assigned are static. This means that you can host services on them. This is where part of the real power comes in. It means you can self-host a Dropbox-like application on your home server and it be available to your entire Tailscale network.

But think bigger – you can even include virtual private servers in Digital Ocean or Azure without an issue.



Trayscale in all its glory. It makes management much easier, especially when using Mullvad VPNs.

It is also possible to make containers part of your tailnet. This does require quite a bit more effort, however, and the process needs to use what's known as a sidecar. A sidecar container provides the network connectivity between the host and the container network.

For the really privacy-conscious aficionados, it is possible to also add Mullvad VPN access to the network. Tailscale works with Mullvad to allow up to five devices to use exit nodes to exit the traffic from the local device to the Mullvad VPN point – see **https://bit.ly/4ejOCbU**.

It only takes 10 minutes to create an account and try it out. If you don't like it, you can uninstall it, but it is genuinely a game changer in terms of ease of use and creating a network that is 'go anywhere' without having to deal with ISP restrictions. LXF

Upon successful completion, the last step is to approve the installation of Tailscale.

HAMMERING UBUNTU FLAT

The unofficial Linux client mentioned above requires Flatpak. This is a simple way to provide apps bundled up into an executable package. No horrible dependencies to fix and they're highly portable (one Flatpak package can run on any Linux with the appropriate Flatpak requirements installed).

Unfortunately, Canonical, thinking it knows best, doesn't support Flatpak out of the box. To make the download work properly, you need to install Flatpak, the GUI plugin and also the repository.

The code below installs everything needed to search for and install Flatpak apps from both the GUI and the command line. From the command prompt, use the following commands:

```
$ sudo apt install flatpak
$ sudo apt install gnome-software-plugin-flatpak
$ sudo flatpak remote-add --if-not-exists flathub https://dl.flathub.org/repo/flathub.flatpakrepo.
```

Once it has been installed, the reader should be able to open the Flatpak package to install Trayscale.

Stuart Burns is a sysadmin for a Fortune 500 enterprise based in that London.

TELLING TALL TAILS

Many people shy away from VPNs and networking, despite potentially huge benefits, such as remote access, sharing files securely and more. Even worse, people often use expensive subscriptions to provide little more than a basic remote desktop, all via proprietary networking software. It is not very GNU-friendly!

The benefits of a VPN are numerous but they can be complex to implement. Tailscale just changed that.

You may have heard about Tailscale. For the uninitiated, it provides a simple and user-friendly way to create a private network layered across all your devices with end-to-end encryption. If you can install an application, you can install Tailscale. It also supports all major platforms, including Linux!

Tailscale transcends the concept of a home network to become a network where it's possible to connect to all your devices, no matter where they sit in terms of physical networks. An abstracted VPN if you will. This abstracted network is called a tailnet. Tailnets are the units of division between users. Each user has a unique tailnet.

No one except you (and your invited users) can see into your tailnet. Everything is encrypted. It even takes care of DNS via a system called MagicDNS. This means there's no more configuring dynamic DNS clients or fighting with routers to allow port forwarding.

What makes Tailscale even better is that the complexity around setting up VPN networks is abstracted away. There are no 'create your certificate authority' instructions. So simple!

MAXIMISE YOUR SEED SPEEDS!

Nate Drake introduces you to one of the internet's best kept secrets to turbocharge your downloads and uploads.

BitTorrent is a protocol for peer-to-peer file sharing. BitTorrent trackers can provide a list of files available for transfer, and allow the client to find peer users, known as seeds, who may transfer the files using a specialised BitTorrent client. Given how efficient it is for sharing large files, it's hardly surprising that BitTorrent accounts for around 3% of all internet traffic.

As BitTorrent can be used to share copyrighted files, some networks and ISPs block or throttle the protocol, making downloading directly to your machine difficult.

This is where seedboxes come in. A seedbox is a remote server, designed for downloading/uploading files via BitTorrent at high speeds – usually 100Mb/s (8MB/s) to 10Gb/s (1,250MB/s). Once the files are in the seedbox, users can download them using other common protocols such as HTTP, FTP, SFTP or *rsync*.

Why do you need a seedbox?

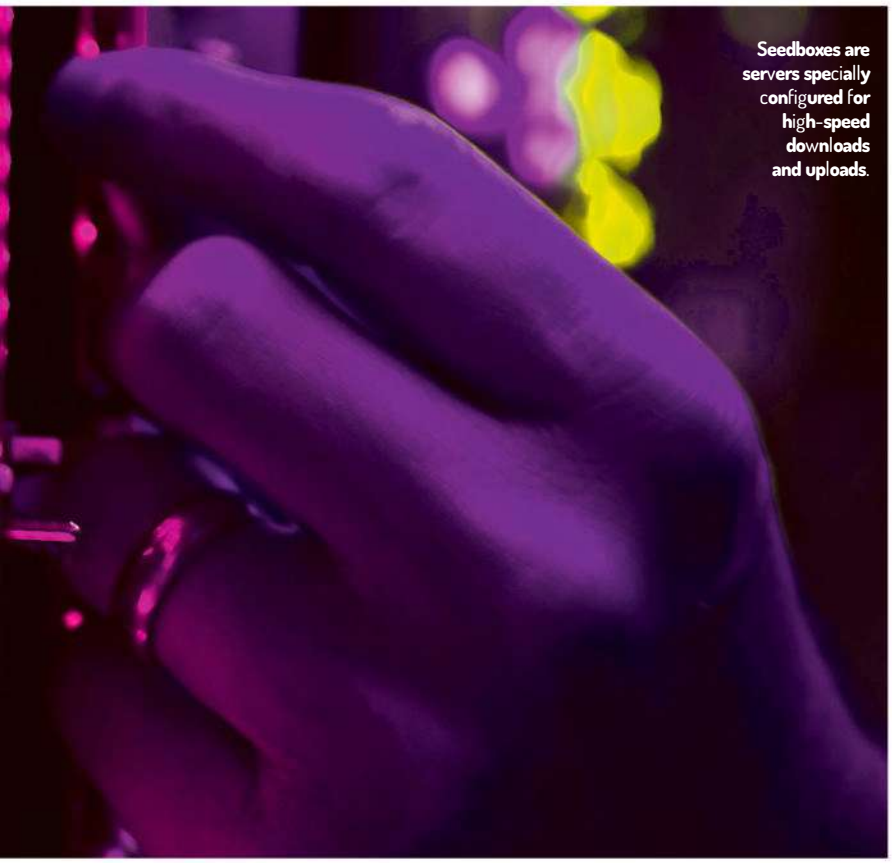
It's true that you can mask your IP to some extent by using a VPN, though not all allow BitTorrent traffic and you shouldn't use it over Tor. This also won't help if your ISP places caps on how much data you can download. Some seedbox providers let you install *OpenVPN Server,* allowing you to connect securely without paying for a separate subscription.

Aside from being designed for fast file sharing, all seedboxes come with a number of preinstalled apps. Chief among these are popular BitTorrent clients like *ruTorrent, Deluge, qbittorent* and *Transmission.*

Setting up and configuring these via the seedbox web interface works virtually identically to desktop clients, with the caveat that they can be connected 24/7. This makes it easier to maintain a good sharing ratio, which is necessary for some private trackers.

Upon registration, your seedbox offers various ways to access files you download to it. Certain providers may offer a way to do this via HTTP, such as through a server app like *Filebrowser.* You can also always access the seedbox via FTP, as you would with any server, but for security reasons, we recommend using a service that supports SFTP via a reliable client like *FileZilla.*

Naturally, copyright laws vary by jurisdiction, so laws may operate differently where you live from the area



Seedboxes are servers specially configured for high-speed downloads and uploads.

CREDIT: Getty Images/ Bill Hinton Photography, Giga-Rapid/Nate

where your seedbox is based. We suggest taking legal advice before sharing or downloading any files.

Having decided that a seedbox is right for you, we encourage you to research providers to find one that fits your requirements. Popular services include Whatbox (**https://whatbox.ca**), Seed Host (**www.seedhost.eu**) and GigaRapid (**https://giga-rapid.com**), which we've used for the screenshots in this guide.

When signing up with a provider, remember that it's less important where the seedbox service itself is based than where its servers are located. For instance, while Whatbox is based in Canada, it has servers in the Netherlands, which can considerably reduce latency for European users.

Set up your seed box

Assuming you've selected a particular seedbox provider, your first step should be selecting a plan that's suitable for you. This is another great perk of using a seedbox versus a home server setup, as you can choose a low-cost, low-spec plan, then upgrade when you need better features.

Given that both seedboxes and end users have different needs and capabilities, we recommend starting out with a low-priced plan, and ideally a provider that offers a free trial. Sometimes this is easier said than done, as the lowest pricing tiers are often sold out. Your chosen plan should rest on three main criteria that we'll cover here:

1. Location: As discussed, however fast data downloads to your seedbox, you need to access it from one of your own devices at some stage either through direct download or streaming. Make sure you check exactly where the server is hosted to reduce latency.

SEEDBOX + VPN

As your seedbox is effectively a server, many providers offer plans that support running your instance as a VPN server. Take the time to check through the Applications section to discover popular software such as *OpenVPN Access Server* and *WireGuard.*

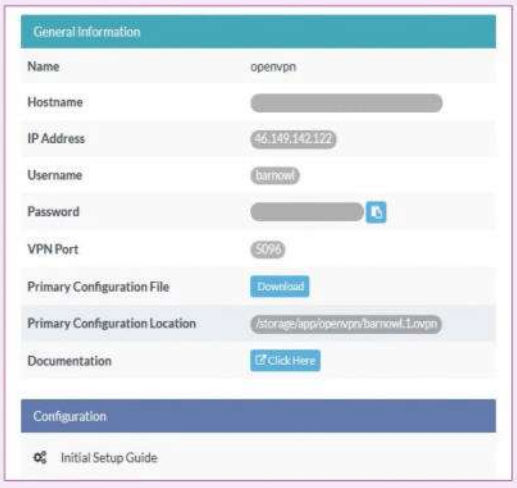
While generating public/private key pairs to establish a secure VPN connection isn't difficult via the command line, most seedbox providers save you the trouble by setting up a subdomain to connect via your client, as well as a suitable . OVPN configuration file, in the case of OpenVPN.

The main advantage of connecting to your seedbox via a VPN is that as your traffic is encrypted, anyone with access to your ISP's record won't know that you've connected to your seedbox, nor what type of content you're accessing, such as streaming video.

Using a seedbox like this comes with some drawbacks relative to using a dedicated VPN service, however. First, VPN usage may count towards your data limit. If you decide to browse the web while using your seedbox as a VPN server, remember that its IP address is visible, so can be targeted by bad actors, such as for denial of service attacks. Check your provider's privacy policy for what information it logs .

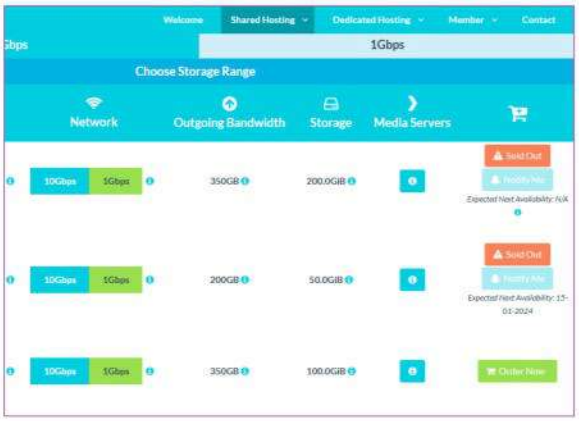
Once the VPN configuration is complete on the seedbox side, we recommend using the accompanying open source project *OpenVPN Connect* (**https://openvpn.net/client/**) for VPN usage on your own devices.

If your ISP and/or local region block VPN connections, check whether your seedbox provider's OpenVPN implementation is v2.9 or later. If so, check if you can configure TLS control channel security in the Admin web UI. From here, you can enable *tls-crypt,* which encrypts data packets and the TLS control panel to make it resemble ordinary HTTPS traffic. This makes it harder to detect and block.



If your seedbox supports OpenVPN Access Server, you can usually install the configuration files with just a few clicks.

Unless you have an unlimited plan, keep a close eye on bandwidth usage. Most providers enable you to break this down on a per-app basis.



Read through and compare seedbox plans. If available, find a low-cost trial before paying for a seedbox with loads of bandwidth and storage.

Some providers, such as Whatbox, offer a multi-threaded speed test on their website (**https://whatbox.ca/speedtest**).

Specifying the location is also important if you want to share files legally, because some countries, such as Spain, the Netherlands, Switzerland and Mexico, have more relaxed copyright laws. Again, make sure you check with a local lawyer before diving in.

2. Bandwidth: Seedboxes offer various download and upload speeds, but they're typically around 100Mbit/s. This means that all things being equal, a 1GB file could download to your seedbox in under 30 seconds. That file can be uploaded to other users in the same time, creating a 1:1 share ratio. This gives you an advantage over sharing from home, as most domestic connections offer minuscule upload speeds relative to downloads. If you're using a shared network and/or drive, your transfer speed can be affected by other users' activity. Some providers offer plans with bare-metal servers/dedicated drives, but these cost more.

Some seedbox services offer 'unlimited' plans, but most have a traffic quota. In other words, there's an upper limit on how much you can download and upload. Check your plan policy carefully – many providers don't count downloads from your seedbox via FTP towards this limit. Certain bundled apps, like those that allow HTTP downloads, may also be excluded.

Make sure you check your seedbox's policy for users who exceed their quota. In most cases, it means your traffic is capped. For instance, if you exceed your bandwidth allocation for any GigaRapid plan, your download speed is reduced to 20Mb/s, and upload speed to 10Mb/s. Most providers allow you to purchase additional bandwidth if you exceed your limit.

3. Storage: If you plan to use your seedbox to store large files, you need a generous amount of storage. Given the blazingly fast download speeds of some services, even a 1Gb/s seedbox can find its download speed limited by the write speed of the disk. If this is a concern, consider using a service such as Whatbox, which offers NVMe SSDs. You get the best read/write speeds from a dedicated server with its own drive, but these plans are more expensive. It's better to start with a plan using a shared drive, then upgrade as necessary.

Software clients

If you're signing up for a seedbox, it's likely that you already have a favourite BitTorrent client. Still, not

» KEEPING IT SECRET WITH SIGNUP SECURITY

There's nothing illegal about renting or using a seedbox, but if it's based in a country with particularly robust IP laws, it could be targeted. In March 2023, a seedbox provider in Denmark was targeted by the local government, resulting in the prosecution of the owner, staff members and even some users.

While you may only use your seedbox to download and share lawful content, the less personal data you give to your provider, the less likely you are to be targeted by spurious legal cases. This is a particular concern if you share a seedbox with others, as if someone misuses their account, their activity could be confused with yours.

Fortunately, most seedbox services actually only collect a minimum of information. If you don't feel comfortable giving out your personal email address for seedbox registration, consider generating a temporary disposable one using a service such as GuerillaMail (**www.guerrillamail.com**).

The provider may also ask for your name and address, but during our tests with GigaRapid, the site happily accepted our PO box instead of a street address.

When paying, you should also check if your chosen provider accepts anonymous payment methods such as Bitcoin. The Seedbox Guide website maintains a list of current providers (**https://seedboxgui.de/seedbox/**), which you can filter by payment method.

Make sure you check your seedbox provider's privacy policy carefully for what personal information it logs when you sign in. If it does record the IP addresses of visitors, consider connecting via a third-party VPN service. If you do this, all connections are routed via the VPN server so that only its IP is logged, rather than that of your home devices.

Remember, if you use your seedbox service to host any personal content, such as family photos, your privacy could be at risk if the server is hacked or seized. You can reduce this risk by securing files using client-side encryption – check whether your seedbox provider offers applications such as *Nextcloud* that support this.

CREDIT: Giga-Rapid/Nate

every seedbox plan offers every app listed on the main website, so make sure you check the policy carefully. Many providers also only allow you to install and launch one BitTorrent client at a time, so choose wisely.

Server apps

A seedbox is essentially a customised server, so can be used for a great deal more than just downloading and playing media. Keep in mind the range of torrent clients on offer and if you feel like expanding its functions, *Nextcloud* should be high on your list as it can provide a host of document handling features.

Root access

If you like to customise your server with your own programs and perform manual updates/upgrades, you may want to choose a seedbox plan that offers root access. This is usually unnecessary, given the number of bundled apps. There's also a security risk in having root access, given that in the wrong hands it can be used to access your private files.

If your chosen seedbox provider doesn't offer root and you do need to run a program that's not currently listed in the available applications, you may be able to compile and run it via a local account. Contact your seedbox provider's support page for help with this.

Dicing with the dashboard

Assuming you've chosen and enrolled for a seedbox plan, it's time to set it up according to your tastes. The exact configuration of each seedbox service is different, so you should check your provider's support page for specific steps.

Still, it's likely that upon login, your seedbox provider takes you to your main dashboard. From here, you can usually see a list of any active seedboxes. Clicking on the name of an individual plan/seedbox also usually displays a helpful infographic indicating CPU, bandwidth, memory and disk usage. Keep a careful eye on these to make sure you stay within the assigned quota and avoid speed caps.

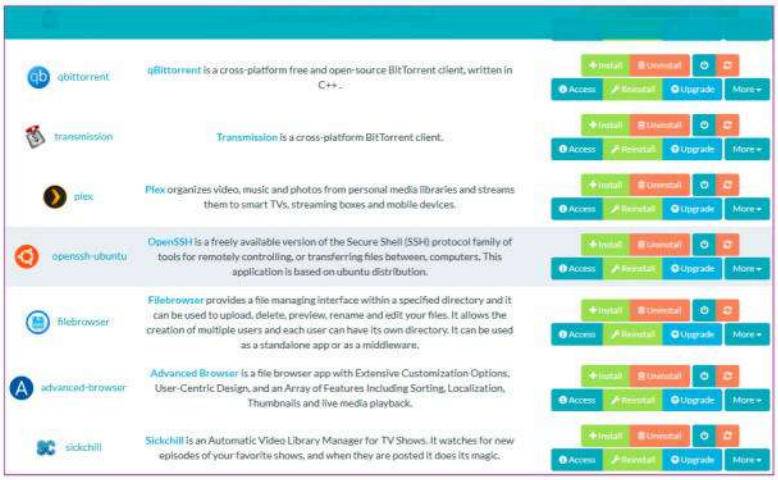
If this is your first login, you may be asked to specify a username and password. The main dashboard should indicate the IPV4 address of the server, along with the relevant ports for FTP/SSH access. You may need to click to reveal these and/or re-enter your password.

The dashboard is also a good place to go if you mess up your configuration, such as forgetting passwords or installing the wrong apps. From here, you can wipe your seedbox disk/credentials and start over.

Configure your client

Having mastered the dashboard, it's time to start using your seedbox for its intended purpose. Again, the exact steps you need to take in order to set up your chosen BitTorrent client vary, but it's likely that you can view a list of available apps or installers via the dashboard.

You most likely need to click Install via the UI to actually download and set up your chosen package. Your seedbox will also ask you to generate an app-specific password for the torrent client or set one yourself. You can usually view this on the app's info screen. You're prompted for it upon launch. Even if you've used the BitTorrent client before, take some time to go through the web UI to note any differences



– in particular, open Preferences to double-check the exact location of downloaded files.

It's now time to take your client for a test spin. Find a suitable torrent file from the public domain – for example, via **archive.org** – then add its URL. Remember that each time you share this file with a peer, it can count towards your traffic limit, so make sure you set a sharing ratio – 2, for example. Consult your BitTorrent client documentation if you need help.

Most seedbox services offer a handy list of installable applications, saving the trouble of tinkering with the command line to download packages and dependencies.

Access your files (http/s)

One of the quickest and easiest ways to access files downloaded via your seedbox is via a specialised app, such as *Advanced Browser.* Take some time to browse the available apps in your seedbox, and go through the install process as you did for the BitTorrent client.

On first launch, you may also be asked for an app-specific password. Most apps of this kind allow you to simply click on a file stored in the seedbox to begin the download, though some, such as *Advanced Browser,* can also play media content in your browser window.

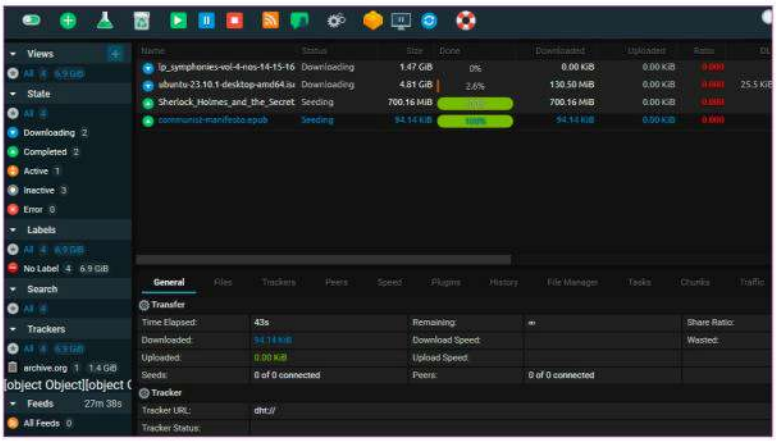
Aside from easy setup, the main advantage of accessing your files this way is that you can access them securely via SSL – the connection is secured by the TLS certificate issued by your seedbox provider. However, most such programs also don't allow you to move or edit files, which makes it impossible to effectively manage your media collection.

Manage your files (ftp)

Upon first login, it's likely that you'll have created a username and password. The main dashboard should also list the specific FTP port for your seedbox. As a *Linux Format* reader, it's likely that you've been using FTP since you were knee-high to a grasshopper, and we have no desire to patronise you.

Still, not all seedbox providers implement FTP in the same way. If you've chosen a seedbox for privacy reasons, remember that plain FTP is unencrypted. This includes your username and password, so if your provider only offers access this way, anyone monitoring your connection will know exactly what you're downloading, and can even harvest your login credentials. For this reason, we recommend finding a seedbox provider that supports SFTP or FTPS, which can use TLS encryption for file transfers.

It's also very unlikely that your seedbox instance uses the traditional TCP port 21 for FTP transfers. If



Seedboxes offer versions of most popular BitTorrent clients, including ruTorrent, Deluge, qBittorrent and Transmission.

you're setting up an FTP connection manually, check your seedbox dashboard to determine the port used. Unless you already have a preferred program, we recommend the free and open source *FileZilla.* This FTP client is such a popular choice that our chosen seedbox provider, GigaRapid, had an automatic *FileZilla Site Manager* configuration file available for download.

When configuring connections in *FileZilla*, it's a simple matter to choose a secure method under Encryption, such as Require Explicit FTP Over TLS. Check your seedbox provider's help pages to determine which are supported.

On first connection, be sure to carefully check the certificate fingerprint and the listed domains to make sure they exactly match those of your seedbox before choosing to trust it. Accessing your seedbox data in this way makes it much easier to manage, as unlike with the HTTP browser, you can edit names, move files and create new directories. When renaming files in your FTP client, don't fall into the same trap that we did by accidentally deleting the file extension.

If you plan to use your seedbox as a media server, this may be a good time to create a dedicated folder for content in the root directory – call it something like **plex-movies**. If you plan to seed a file, now is the perfect time to use your FTP client to upload it. Naturally, you should make sure you upload to a folder to which your BitTorrent client has access.

Track bandwidth usage

While the main dashboard can give you a quick overview of bandwidth usage to make sure you haven't exceeded your quota, you can usually click into this section to view more detailed stats. This enables you to look beyond the simple number of GB received and sent, and into more detailed data, such as how much of this data is exempt – for example, because it was due to FTP transfers – as well as examine data usage on a per-app basis.

Manage your media

If you want to avoid the trouble of downloading media files in your seedbox to play at home, most providers support media server software such as *Plex* or *Jellyfin.* This offers a few advantages over a domestic setup. Firstly, if you enjoy travelling, you can take your media server wherever you go by accessing your media server using a portable medium, such as an Amazon Firestick.

Secondly, if you want to share your downloads with others, you can do so without having to open up ports on your home router and share your own bandwidth.

You can find available media server software in the Applications or Installers section of your main dashboard. However, some providers, like our chosen seedbox, GigaRapid, may require you to only run one media server at a time. If you've not done so already, you should first connect to your seedbox via SSH or FTP to create dedicated folders for your media libraries. Ideally, these should be outside your torrent client's download folders.

For our test seedbox, we created two empty folders for *Plex* on the root drive called **plex-movies** and **plex-tv** to store movies and TV shows respectively. Most clients allow you to specify where to move a file once download is complete when you first add a torrent. In *ruTorrent,* for instance, you can right-click and choose Save To.

If you plan to download a lots of movies, you may prefer to streamline this process. The AutoTools feature in *ruTorrent* (accessed via Settings > AutoTools) allows you to sort files according to predefined labels.

Plex media server

When using *Plex* on a seedbox, you also need to enable remote access for your clients to connect. Once the app is installed, you can usually access its configuration via the dashboard. From here, you can note down the designated web port – 57688, for example.

Next, launch the application and create your *Plex* account (if necessary). Once you're signed in, you are asked to set a memorable name for your *Plex* server. Be sure to tick the box marked Allow Me To Access My Media Outside My Home. You can then click Add Library to create your first media library. From here, you can navigate to the folders you created earlier to store your content – for example, **plex-movies**.

Once you're done, click the settings (spanner) icon at the top-right. Select Remote Access under Settings in the left-hand pane, then Show Advanced. From here, you can choose Enable Remote Access. Tick the box next to Manually Specify Public Port, then enter the Web Port number you noted down earlier under Applications for *Plex.*

One of the reasons we favour *Plex* is because the platform has teamed up with Let's Encrypt to support Secure Server Connections. This means that if a snooper were to perform DPI on your internet traffic, they may recognise that you're using *Plex,* but won't be able to decrypt packets to determine which specific content you're streaming.

If you choose Network under Settings, then select Show Advanced once again, you can change Secure Connections from Preferred to Required. This blocks insecure connections, but may also mean that you can't stream media server content via certain devices or applications.

Plex's support pages provide a full list of compatible apps (**https://bit.ly/lxf321plex**) such as Android TV. By default, the local *Plex* web app itself loads over insecure HTTP. If you installed *Plex* via a seedbox,

however, it's likely that the connection is secured by your seedbox domain's SSL certificate.

Jellyfin alternative

As powerful and easy to set up as *Plex* is, you may need to pay to access certain features like Plex Pass to stream content on certain devices. This is why almost all seedbox providers offer plans incorporating *Jellyfin.*

This free and open source media server software may be more limited in scope, but it's also more lightweight. As FOSS, it's also easier to verify that *Jellyfin* has greater respect for user privacy, being free of ads and trackers and it won our *Roundup* this issue.

If you wish to use *Jellyfin*, you only need to select it from your seedbox's Applications or Installers. This launches an extremely intuitive wizard, which asks you to specify your language, username and password.

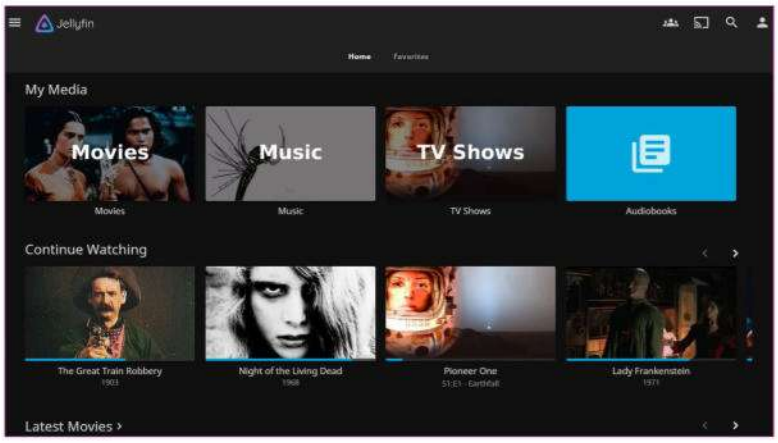
During setup, you come to the Configure Remote Access screen. Make sure you tick both Allow Connections To This Jellyfin Server and Enable Automatic Port Mapping. Once initial setup is complete, navigate to the Info or Settings section for *Jellyfin* from your main dashboard. From here, you can view both the HTTP and HTTPS port numbers.

Return to the *Jellyfin* web UI and click on the icon at the top-right. Select Dashboard under the Admin section. Next, go to Advanced > Networking. Fill in the Public HTTP Port Number and Public HTTPS Port Number fields with the values you noted down earlier. To add your chosen media folders, return to the Dashboard, then select Server > Libraries > Add Media Library.

Sharing seedboxes

The majority of seedboxes are shared – each user has their own account to access a shared server. Resources may be allocated equally or divided between users based on their plans.

The specifics of your seedbox setup vary based on provider and plan, but a shared seedbox is usually sufficient for most users who just want to download/share content and stream media. Even if you don't get the full resources of the server or its entire download speed, applications such as *ruTorrent* and *Jellyfin* are extremely lightweight.



Another popular (if slightly dearer) solution is to give each user their own VPS running on a physical server. This gives you a better guarantee of privacy, plus in most cases you have root access, so can customise it further. Seedbox plans that use VPS are also easy to upgrade if you need more storage or bandwidth.

Naturally, the very best privacy, performance and speeds are to be had from renting entire physical servers. Some providers, such as SeedHost, offer this, but naturally these are the most costly subscriptions.

Seedboxes also make for excellent media servers using installable software such as Plex or Jellyfin. This means you don't have to redownload files to your home computer.

Seedboxes in summary

The decision on whether or not to use a seedbox depends entirely on your needs, your ISP and your resources. Power users may well be happy to take the time to set up their own web server with PHP, then install *rTorrent, ruTorrent,* various dependencies, configure RPC socket and so on to get it running. Alternatively, you can pay a few pounds to rent a seedbox and have your chosen BitTorrent client running out of the box.

If you're considering setting up your own seedbox to do more than just download files, we encourage you to read the plan carefully, because not every application is supported for every plan. LXF

» RATION YOUR RATIOS

Most seedbox providers offer *ruTorrent,* a web-based version of the text-based *rTorrent* client, because it's easy to manage downloads via the web interface (which itself is based on *uTorrent's* web UI) and it's also highly customisable.

Once *ruTorrent* is installed, set file-sharing ratios. Start by clicking on the settings icon, then choose Ratio Groups in the left-hand pane.

Each ratio group has four different conditions (Min%; Max%; UL,GiB and Time,h) and an action to take when these conditions are met. By default, this is to stop sharing the file.

For instance, if you simply want to stop sharing after a ratio of has been met, like 2.5, just set Min% and Max% to the same value, such as 250%. Most private trackers require maintaining a certain ratio, so this is all you need to configure.

You can also use the Default Ratio Group drop-down menu at the bottom-right to make sure any torrents added moving forward conform to the conditions of a particular ratio group.

If you don't want to use simple sharing ratios, UL,GiB can be used to specify an action when a certain amount of data has been uploaded. For example, if you wanted to stop a torrent seeding after 10GiB have been uploaded, regardless of ratio, you could input:

```
Min% : 99999 Max%: 0%
UL,GiB : 10 Time,h: -1
Action: Stop
```

Some private trackers require that files are shared for a certain amount of time. You can specify this (in hours) via the Time,h condition. Note that by default, this is set to -1 (which is unlimited).

For instance, if you wanted to stop a torrent seeding after 48 hours, regardless of the ratio or how much data has been uploaded, you could input:

```
Min% : 99999 Max%: 0%
UL,GiB : (default value)
Time,h: 48 Action: Stop
```

HotPicks

Snoop » Nuclear » Media Downloader » Concessio » Endless Key » Tuta » FireDragon » AntiMicroX » LibreQuake » nnn » Micro

Mayank Sharma thinks it's time that compiling *HotPicks* is added to the list of activities that should be an Olympic sport but aren't.

SEARCH IN FILE

Snoop

Version: 0.4.0 **Web:** https://gitlab.gnome.org/philippun1/snoop/

You can use the multifaceted *Grep* utility to search for text within files, but if you are averse to command-line utilities, you can use *Snoop* instead. It isn't as dexterous as *Grep,* but *Snoop* can be used to search for text or a string of text inside files. *Snoop* is available as a Flatpak and can be installed with `flatpak install flathub de.philippun1.Snoop`.

If you've used a search utility before, *Snoop's* interface shouldn't pose a challenge. There's a text box at the top where you specify the text you want to search. The pull-down menu next to the search box has a list of the previous search terms. As you'd expect, it's empty on the first run.

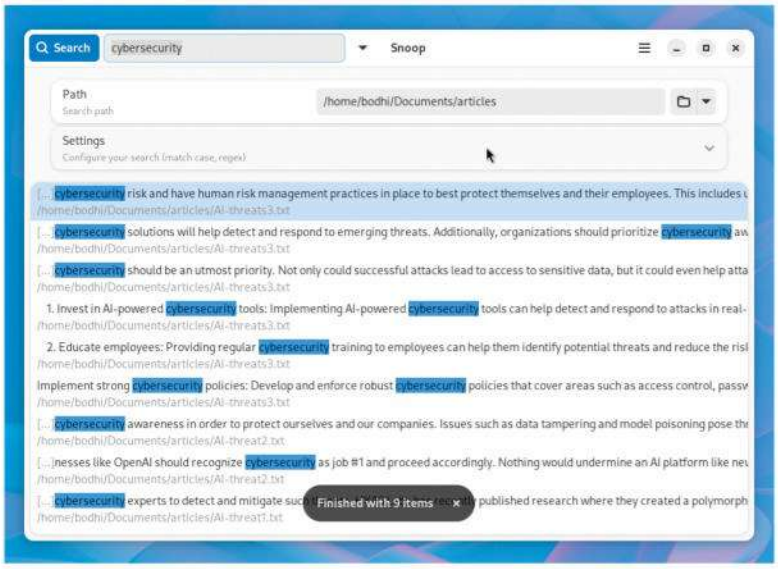
After specifying the search string, head to the Path option and point the app to the folder you want to search for files with the specified string. You can either specify the path manually or use the folder icon to navigate the filesystem to point to the path graphically. Again, the pull-down menu lists previously used paths, and is initially empty.

Then roll down the Settings options. None of the options are selected initially, and it's perfectly normal to run the search without selecting any options.

By default, all searches are case-insensitive. However, you can override this behaviour by selecting the Match Case setting. When toggled, the app only includes results where the case matches that of the string specified in the search box.

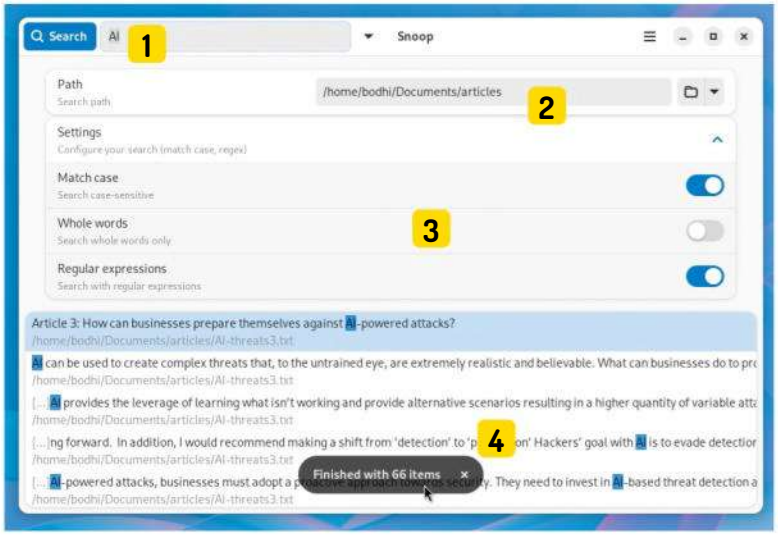
Similarly, if you enter multiple words in the search string, *Snoop* displays results where it finds any of the words inside a file. You can override this behaviour with the Whole Words setting, which ensures the application only brings up those files where the entire string appears in the file, instead of just a word.

After constructing your query, hit Search to let *Snoop* find the specified string in the highlighted location with the toggled settings.



While you can use Snoop on any desktop, Gnome users can embed it in the file manager by installing the Nautilus extension from the Preferences dialog.

LET'S EXPLORE SNOOP...



1 Specify **q**uery
Enter your search string in the space provided. The app also keeps a record of your last 10 search **q**ueries.

2 Set path
Use this area to specify the search path. The app then searches files under the search path for the search string.

3 Toggle settings
Snoop has three very pertinent options (match case, whole words, regular expressions) to make the search strings more useful.

4 Results
Snoop displays the line number in the file that matches the search **q**uery. Double-click a result to open it in the built-in text editor.

MUSIC PLAYER

Nuclear

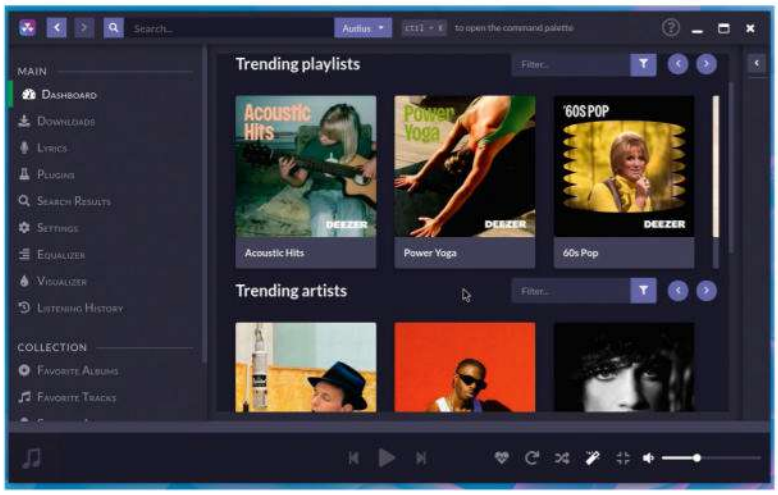
Version: 0.6.38
Web: https://nuclear.js.org

Nuclear is an interesting music streaming application in that it's designed to pull all kinds of content, from all kinds of free online sources. This means that you can search for tracks, artists and albums, and the software goes and finds information about them, and any related info such as lyrics, along with song streams. *Nuclear* also takes pride in the fact that it does all this without pushing ads.

The app recommends using its AppImage, but it is also available as a Flatpak, Snap and more.

Nuclear has a clean and simple user interface. There's a search bar at the top, which searches for and streams music from YouTube, by default. When it finds a track, you can click on it to play it right away or add it to the queue or a playlist. You can also mark the track as a favourite, and even download the track if the provider allows it.

Nuclear has three kinds of providers: streaming, metadata and lyrics. They are shown in the plugins section, listed in the sidebar menu on the left.



Nuclear shows a list of trending playlists, artists and top tracks on the main dashboard. You can also browse music though genres.

Streaming providers are where the actual music comes from. The default is YouTube. When you add a track to the queue, the streaming provider tries its best to find a matching stream. Besides YouTube, the app has about half a dozen other providers, including SoundCloud, and Jamendo.

Lyrics providers are only used to fetch the song lyrics. The app only supports a couple of providers, and if it finds the lyrics, you can view them from under the Lyrics option in the sidebar.

Besides streaming music, you can also use *Nuclear* to play offline tracks in your library, from under the Local Library option. Head to Listening History to scroll through a list of recently played tracks.

DOWNLOAD ONLINE MEDIA

Media Downloader

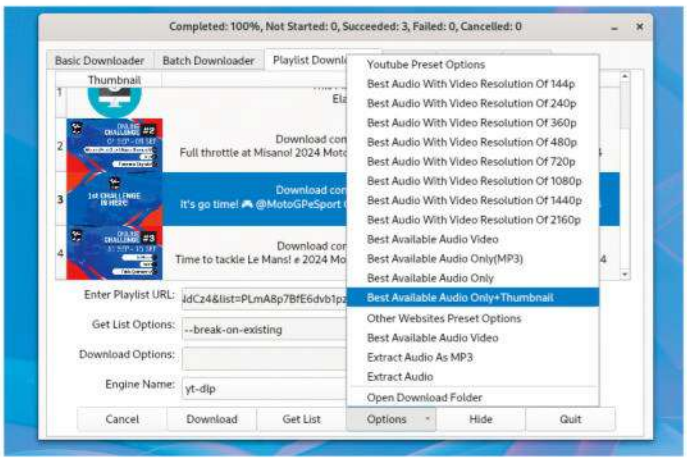
Version: 5.0.1 **Web:** https://mhogomchungu.github.io/media-downloader/

Sure, there are tons of media downloaders, and we've featured a few of them, but there's no one app that's as comprehensive as the unimaginatively named *Media Downloader,* a GUI front-end to multiple CLI-based tools that download media.

The app is distributed on Flathub and can be installed with `flatpak install flathub io.github.mhogomchungu.media-downloader`.

The *Qt*-based app isn't a looker, especially on non-KDE desktops, but is fairly lightweight and quick off the blocks. On startup, it checks the installed versions of the CLI tools it uses to get things done, and auto downloads any updates. The app has instructions on how to disable this check to shave off a few seconds.

By default, the app uses *yt-dlp,* a feature-rich CLI-based audio and video downloader. But it also supports other downloaders. For instance, there's *gallery-dl,* which helps download media from Instagram. There's also *wget,* which you can use to download ISO images and other large files.



Use Media Downloader's Download Option pull-down menu to select one of the several preset YouTube options.

The downloaders are implemented as plugins. You have to manually grab a plugin's JSON file, then point the app to it by heading to Configure > Add A Plugin. Once it's installed head to Basic Downloader and select the downloader from the Engine Name drop-down list.

If you are using *yt-dlp,* the app offers two options to download multiple videos at once. First up, there's the Playlist Downloader. As you'd expect, this downloads video playlists. Paste the URL of the playlist here, and click Get List to fetch the list of videos. Once done, click Download to grab the videos. Then there's the Batch Downloader, where you can enter multiple URLs to grab several pieces of content in one go.

LEARN FILE PERMISSIONS

Concessio

Version: 0.1.4 **Web:** https://github.com/ronniedroid/concessio

Good knowledge of file permissions is one of the basic requirements of running a Linux distro. Yet, it isn't a skill many Linux users spend time mastering. *Concessio* will help correct this wrong. You can use the app to get to grips with Linux file permissions.

The app is available on Flathub and can be installed with `flatpak install flathub io.github.ronniedroid.concessi` . It has a minimal but well-designed interface.

Concessio provides a convenient way to convert between symbolic and numeric representations of UNIX file permissions. For instance, you can enter a numeric permission, such as 664 (*not 777–Ed*), in the space provided under the Numeric heading, and press Enter. The app then automatically converts it to the equivalent symbolic representation (which is rw-rw-r, in this case).

In case you are a total newbie, you can use the read, write and execute toggle buttons to specify permissions for the three user groups: user, group and others. Depending on your selection, the app then automatically converts them to the correct numeric and symbolic values.

While you can construct permissions with Concessio, head to the app's Help to read and understand the **UNIX** permissions system.

You can also point the app to a file, and it reads and displays its numeric and symbolic file permissions. This is one feature that both new and experienced users will appreciate. On the downside, you can't use the app to change the permissions of a file on your system. Perhaps this will be added in a future release.

But for now, you can generate any file permission as per your preference. Once you have the necessary permission, you then have to use the copy button to copy the permissions to the clipboard. From here, you can paste the permissions to an external app or utility, such as the CLI-based *chmod* command, to actually change the permissions of a file.

EDUCATIONAL PLATFORM

Endless Key

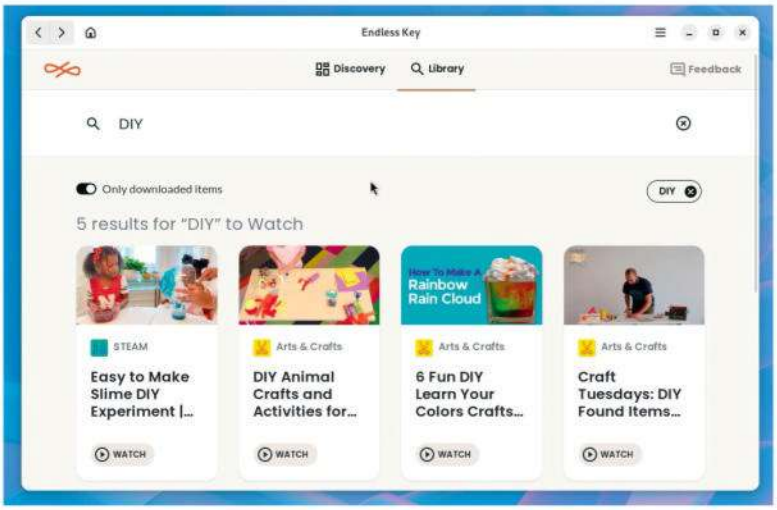
Version: 0.9
Web: www.endlessos.org/key

The *Endless Key* app is an educational platform that claims to provide equitable access to learning resources. It attempts to bridge the digital divide by providing users with access to a major part of its educational resources without relying on an active internet connection.

The app is available as a distro-agnostic Flatpak and can be installed with `flatpak install flathub org.endlessos.Key` .

When you launch the app for the first time, you're asked to select from one of the six available profile options, such as Artist, Scientist, Athlete and more. *Endless Key* then downloads the associated Starter Pack. This initial download is quite bulky in size, but can then be accessed without an internet connection. When it's done, click Show Me to get started and explore the content.

Besides the availability of offline content, the app also takes pride in its curated, hand-picked content for all its supported streams, from reputable sources including Ted-Ed, Khan Academy and SciGirls.



The results page in Endless's Library does have a toggle that only displays content that's already available for offline viewing.

We have mixed feelings about the interface. It's straightforward for us, but could confuse young users.

To start, select a channel from those listed under the Discover tab, including ones that will help you learn a new skill and others that'll help satiate your curiosity.

If you want to search for something specific, switch to the Library tab. You can now search by entering keywords in the search box or by selecting a channel. Remember: this tab lists all channels, including those that aren't yet downloaded to your device. Click the Download button to fetch new content if you are online. There's no way to track the progress of the download, but *Endless* marks it with a green arrow when it's done.

EMAIL CLIENT

Tuta

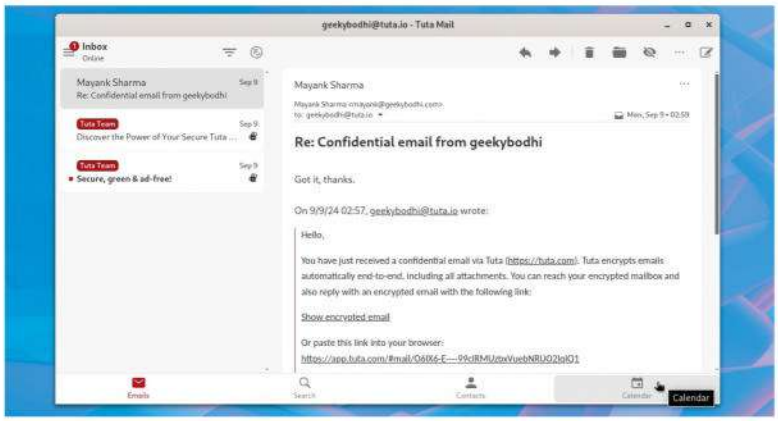
Version: 244.240903.0
Web: https://tuta.com

Tuta is the client for the privacy-focused email service of the same name. The app is available as an official AppImage, but there's also an unverified Flatpak on Flathub. Grab the AppImage and make it an executable with the `chmod +x` command or via the file manager.

When you launch the app, you're asked to log in to the service. If you haven't yet registered with the service, you can use the app to do so. *Tuta* has a tiered payment plan, with a feature-limited free option you can use to try the service.

The service offers a selection of several domains – there's **tuta.io**, **tutamail.com** and others. However, to use **tuta.com**, you have to be signed up to a paid subscription plan. As part of the account creation process, Tuta asks you to jot down a 64-character recovery code. This helps reset your password in case you ever forgot it.

Unlike most 'free' services that harvest your data for advertising purposes, *Tuta* pitches itself as an encrypted email provider. It doesn't have adverts,



and it enables users to send end-to-end encrypted (E2EE) email. In all fairness, *Tuta's* E2EE emails work best when exchanged between *Tuta* users. You can still send encrypted emails to users of other platforms, but they appear very spammy. If you send encrypted mail to an external unencrypted email platform, the recipient receives a link to a temporary *Tuta* account to read the email securely.

You can, of course, still send unencrypted emails using the app, and that's still better than sending one through other services, because *Tuta* promises not to monetise your email.

Every time you launch the app, you get a pop-up asking whether you would like to integrate it into your desktop. Although it's a useful default, you can turn off this check if you find it irritating.

Besides email, the free edition also includes one calendar that you can access from the intuitively laid-out app.

WEB BROWSER

FireDragon

Version: 11.18.1-1 **Web:** https://firedragon.garudalinux.org

FireDragon is a web browser that's focused on protecting your privacy. It's based on *Floorp,* which we've previously featured (**LXF308**). *Floorp* itself bolts on several privacy-focussed extensions atop *Firefox.* But while *Floorp* appears quite similar to *Firefox, FireDragon* looks radically different.

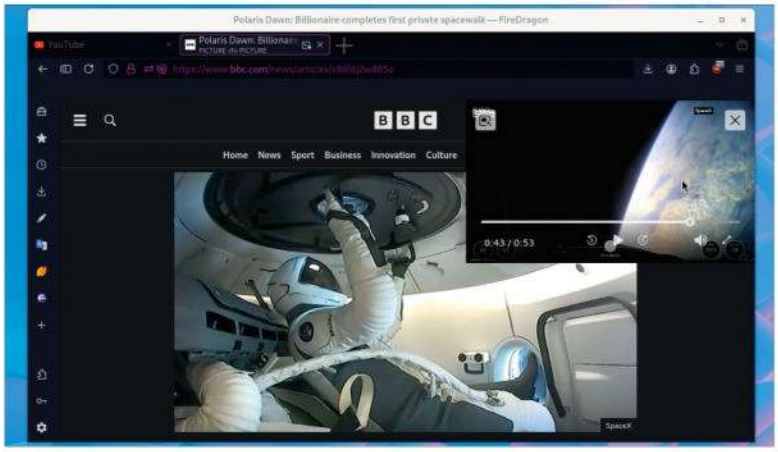
The browser is bundled (and developed) as part of Garuda Linux (**LXF317**), but you can install it atop any distro as it's bundled as an AppImage and Flatpak.

Grab its AppImage from its website and make it an executable using the file manager, or with `chmod +x`.

The browser contains virtually all of *Floorp's* modifications and is stripped of the *Firefox* telemetry code. It also bundles the uBlock Origin content filtering extension by default.

FireDragon uses a dark theme with vibrant colours, called sweet theme. It also enables dark mode for every website, and has a spatter of custom Garuda Linux branding.

When you enter a search term, the browser actually gives you the option to search for it using several



search engines, including DuckDuckGo, Whoogle, Google and several others. Whoogle gets Google search results, but without any ads, JavaScript, AMP links, cookies or IP address tracking

The browser also flaunts its use of *Firefox's* tracking protection tech, called PBMode. It has also tweaked the *Firefox* accounts feature to sync the profile data through its own custom self-hosted sync server (**ffsync.garudalinux.org**).

FireDragon also uses a couple of tricks to load web pages faster. Thanks to its own custom settings, the browser claims to load web pages faster than its other *Firefox*-based competitors. It also includes FastFox tweaks, which help increase any *Firefox* browser's browsing speed.

FireDragon was originally a fork of the LibreWolf browser (**LXF319**), and the developers are still working to integrate its best patches and tweaks in the new base.

JOYSTICK BUTTON MAPPER

AntiMicroX

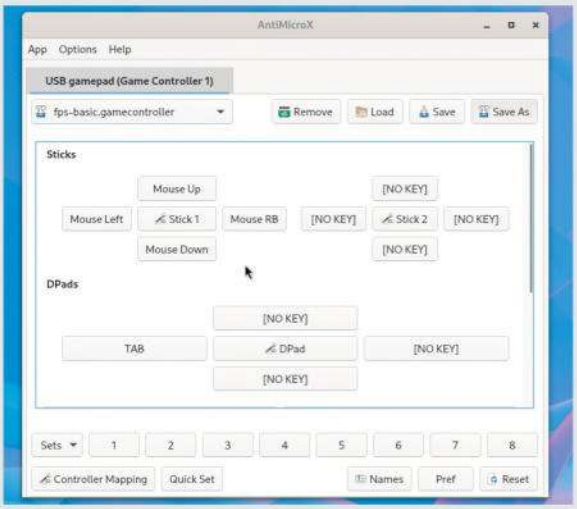
Version: 3.4.1 **Web:** https://github.com/AntiMicroX/antimicrox

Not all games, particularly retro titles, support joysticks and game controllers. Even if they do, many don't work well with these built-for-gaming devices. If you run into such a title, *AntiMicroX* can get your poorly supported joystick or game controller to emulate the better supported keyboard. The app does this by binding keyboard keys and mouse buttons to the sticks and d-pads on your gamepad.

It is available on Flathub and installed with `flatpak install flathub io.github.antimicrox.antimicrox`.

On launch, the app automatically detects your gamepad controller. Linux can detect all kinds of controllers, and even detected our no-name generic game controller. When you press a button on your controller inside the app, the corresponding key lights up on the app's controller map.

To bind the controller buttons to the keyboard button, press the controller button you wish to bind. This lights up the button in *AntiMicroX's* controller map. Use the mouse to click on the button, and then select the action you wish to bind to it.



Besides its usefulness in the case of games without gamepad support by default, you can also use AntiMicroX to control any desktop app.

By default, the app shows the layout for the keyboard. Use the Mouse tab at the bottom to display the keys for the mouse.

For instance, you can bind the left and right d-pad keys on your controller to the left and right mouse buttons. Similarly, you can bind the left, right, top and bottom action keys on the sticks to the left, right, top and bottom arrow keys on the keyboard.

The keys are active as soon as you define them. But to ensure they persist, click the Save button to save your configuration. You can then load them when you restart the app and continue to map the rest of the keys in your controller.

FPS

LibreQuake

Version: 0.07-beta
Web: https://librequake.queer.sh

We're big fans of retro first-person shooters, and *Quake* has a special place in our hearts. Linux users love the hugely popular successor to *Doom,* because of id Software's open stance towards game modifications. The source code of the shoot and scoot title was GPLed, which opened the door for all kinds of enhancements.

Crucially though, id kept the game's maps, objects, textures, sounds and other creative works under its original proprietary licence. *LibreQuake* is working to correct that wrong by creating GPLed game data.

As *LibreQuake* is only game data, you need to pair it with a *Quake* engine, also known as a source port. There's an unending list of these, but the *LibreQuake* project recommends the QuakeSpasmSpiked project. It supports 64-bit CPUs, custom music playback, a new sound driver, and tons of other improvements.

To install the engine, head to **https://triptohell.info/moodles/qss/** and download the latest Linux release, currently 2024-March-01. Extract it with `unzip quake spasm_spiked_linux64_dev -d quakespasm_spiked`.

Despite still being in beta, LibreQuake offers five episodes of playable maps that'll keep you hooked for hours.

Before you can play, you need to download one of the five asset sets. The recommended set is called Full and it contains all of the data needed to play *LibreQuake,* along with mods and other community content. There's also a Lite edition that's designed for low-powered computers.

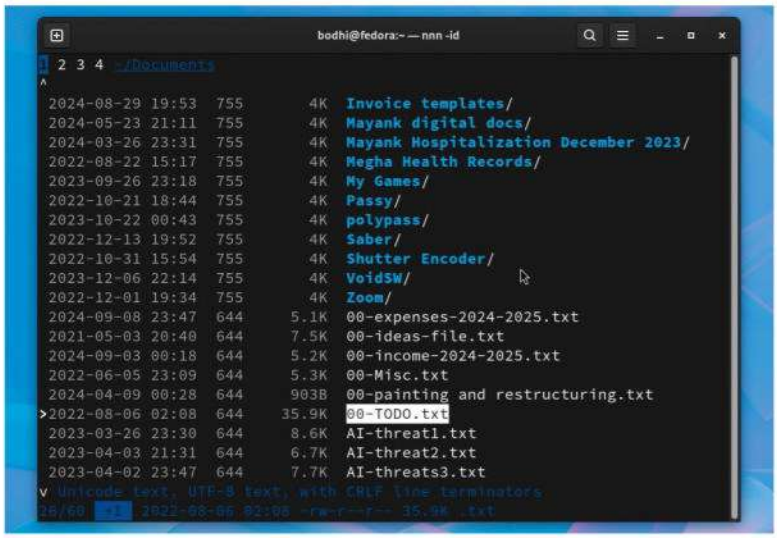
Once you have downloaded the relevant set, extract its contents with `unzip full.zip -d full`. This extracts the **id1** folder, which houses two archives, namely **pak0.pak** and **pak1.pak**, which contain all the game data. Copy this folder into the root of the QuakeSpasmSpiked directory, **quake spasm_spiked_linux64_dev**, and you're good to go.

Now switch to the earlier extracted **quakespasm_spiked** directory, and launch the engine with `./quakespasm-spiked-linux64`.

FILE MANAGER

nnn

Version: 5.0
Web: https://github.com/jarun/nnn



File management utilities come in all shapes and sizes. Mainstream distros usually ship with bulky file managers, while lightweight distros use file managers that don't exert a heavy footprint on the hardware resources. Then there's *nnn*.

Dubbed the "snappy file manager for ninjas", *nnn* works on the command line. Its binary weighs a measly 50KB and the app itself takes up less than 5MB of space in the RAM.

While *nnn* is available in the repos of most desktop distros, the best and pain-free option to install its latest release is via the *HomeBrew* utility. Refer to **LXF303**, page 90, to set up *HomeBrew* in your distro. When it's up and running, use `homebrew install nnn` to grab and install the latest version of *nnn*.

To use the file manager to open the current directory, simply enter `nnn` in the console. This shows a list of files and directories under the current directory.

Press the left key to go to the parent directory, while the right key takes you into the sub-directory. If you press it on a file, it opens it using the distro's default app. If you hit `e` on a file, the app opens it in the *nano* text editor.

You can also use `nnn -d` to start *nnn* in detailed mode. Besides files and directories, it shows their last modification time and date, permissions and size.

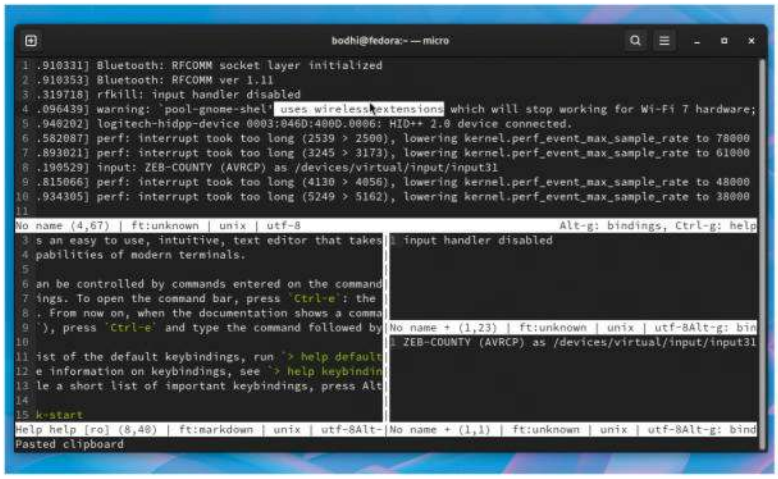
To get *nnn* to show hidden files, use the **-H** option, such as `nnn -H`. Another useful option is **-i**, which displays brief information about the current file, such as whether it's a directory, a PDF, an XML document and such. It can also display the resolution of certain images, such as PNGs and GIFs. Press `?` to view a list of all its keybindings, and `q` to quit.

As it is with **CLI** utilities, you'll have to spend some time perusing through nnn's man page before you can use it proficiently.

TEXT EDITOR

Micro

Version: 2.0.14 **Web:** https://micro-editor.github.io/index.html



Micro is a terminal-based text editor that pitches itself as the spiritual successor of *nano*. It claims to take advantage of the capabilities of modern day terminals, and its developers pitch it as an editor for people who regularly edit files over SSH.

The app comes as a single static binary with no dependencies. Download it with `curl https://getmic.ro | bas`. The script places the *Micro* binary in the current directory. You can run it with `./micro`. It's best to move the binary to wherever your distro houses system-wide binaries, such as **/usr/bin**, with `sudo mv micro /usr/bin`.

Start it by running `micro path/to/file.txt` or simply `micro` to open an empty buffer. You can also use *Micro* to create buffers from stdin, such as `dmesg | micr`, or even `dmesg | tail | micro`.

For a CLI tool, *Micro* has good mouse support. In addition to the keyboard, you can also use the mouse to jump inside the file. You can use the mouse to manipulate the text, too. Simply clicking and dragging selects text. A double-click selects the current word, while a triple-click selects the current line.

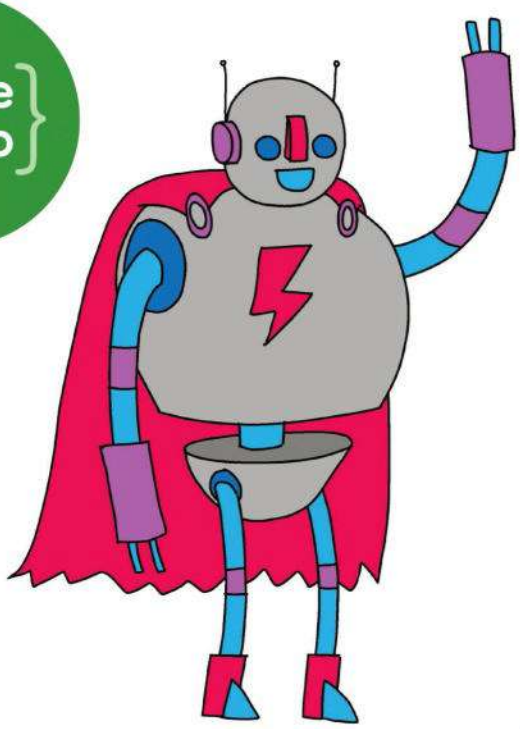
Micro uses common keybindings to copy, paste, undo changes, save the document and more. Alt+g brings up a list of common bindings.

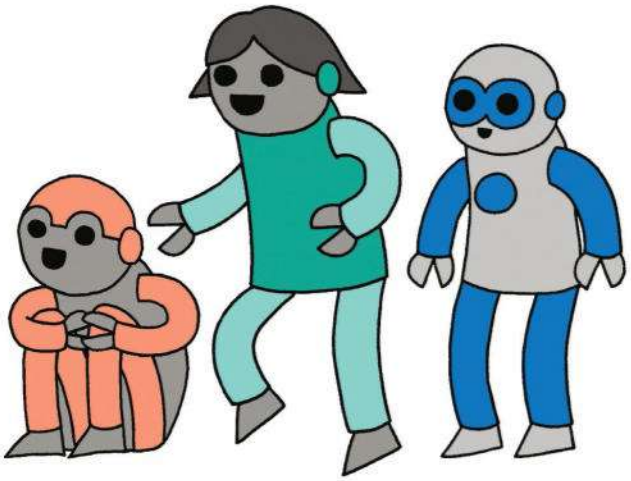
The app automatically splits the interface to display additional information, such as the list of common keybindings.

To see a better example, press Ctrl+e to go to command mode. Type `Help` to view its user manual. This, too, is displayed in a split view. Press Ctrl+w to move between splits. You can also manually split the current window either horizontally by typing `hsplit` or vertically with `vsplit` in the command mode. LXF

Besides being a dexterous **CLI** text editor, **M**icro can also be useful for programmers thanks to its support for syntax highlighting for over **130** coding languages.

code club





Can you help inspire the next generation of coders?

Code Club is a nationwide network of volunteer-led after school clubs for children aged 9-11.

We're always looking for people with coding skills to volunteer to run a club at their local primary school, library or community centre for an hour a week.

You can team up with colleagues, a teacher will be there to support you and we provide all the materials you'll need to help get children excited about digital making.

There are loads of ways to get involved!

So to find out more, join us at **www.codeclub.org.uk**

OSQUERY

Credit: https://osquery.io

Using Osquery to explore your system

Ever-curious **David Bolton** shows how to use the Osquery application to view your system via a series of SQL select queries.

OUR EXPERT

David Bolton is on a quest to leave no byte unturned on his computers, so has installed yet another utility to document the insides of his Ubuntu system with SQL.

Put simply, *Osquery* is software that enables you to run SQL queries to provide information about your system. With *Osquery*, SQL tables represent abstract concepts such as running processes, loaded kernel modules, open network connections, browser plugins, hardware events or file hashes.

The idea is that rather than running lots of different utilities to find out things about your system, you instead run an SQL query on one of the tables. Behind the scenes, *Osquery* has mapped the state of your system into lots of different tables.

How many tables? Well **https://osquery.io/schema/5.12.1/** lets you select your OS type (Linux, Mac and Windows) and shows a clickable list. For Linux, there are 154 tables. A significant proportion of these are tied into software you have installed, so there are tables for *Chrome, Firefox, Docker,* npm packages and quite a few more.

Just click on a table name in the list to see all the fields. When you want to inspect a concept, you 'select' the data, and the associated OS APIs are called in real time.

Ironically, the query `select * from cpu_info;` returns nothing on our system because it's running in a virtual machine, although `cpuid` does. Other queries, such as `select * from deb_packages;` , return many rows. In that case, you might find `select count(*) from deb_packages;` more useful.

To see all the tables, use the *Osquery* command `.tables` . You can then follow it with the command `.schema tablename` to see the fields for the specified table. This produces the SQL **create** table for the specified table. Here, for example, is what `.schema cupid` produces:

```
CREATE TABLE cpuid(
  `feature` TEXT,
  `value` TEXT,
  `output_register` TEXT,
  `output_bit` INTEGER,
  `input_eax` TEXT);
```

The **table** definition **for** cpu_info.

Although *Osquery* supports *SQLite*'s SQL syntax, you don't need to know anything about how to update, insert or delete SQL queries. The tables are read-only, so unless you go into advanced stuff and start creating and populating new virtual tables, you can stick with select queries.

Arguments over tables

Some tables, of which **file** is probably the best-known, need to have a **where** clause for the SQL query to return results. You can identify these tables because in the schemas, some fields have a pushpin icon – for instance, both path and directory fields in **file** (**https://osquery.io/schema/5.12.1/#file**) have pushpins.

SQL wildcards are **%** for any character and **?** for matching a single character. So, if you want to know how many files are under your home directory you'd run these queries.

An interesting example is doing a join between the **file** table and the **hash** table to get the **md5** and **sha256** hashes of all files under your current location:

```
select path, md5,sha256 from file join hash using (path)
where file.directory='%' order by path;
```

QUICK TIP

Some tables have a lot of rows. Use the where clause to restrict the number of rows returned or count(*) to return a count of how many there are. The query `select count(*) from deb_packages;` **returns 1861 on our system.**

However, be careful where you run that because the results depend on the files and folders beneath. There are a few thousand files under / for example.

Using **(column)** is an *SQLite* enhancement for joins. Section 9.2 on the *SQLite* page **www.sqlite.org/lang_select.html** documents it.

Of course there's maths!

The people behind *Osquery* have added various additional SQL functions and aggregations. So, you can use **sqrt**, **log**, **log10**, **ceil**, **floor**, **power**, **pi** and **sin**, **cos**, **tan**, **cot**, **asin**, **acos**, **atan** and radians-to-degrees conversions. There are also some extra hash functions, string functions, regex, network and collations. This *Osquery* page on Introduction to SQL (**https://bit.ly/lxf321intro**) covers everything.

Daemon or shell?

These are two independent systems *Osqueryi* (shell) and osquerysd (daemon). So far, all the examples we've used here have used the shell – however, for an admin, the daemon comes in a lot more useful. There's a lot more to it, as well.

The daemon is there to monitor the host upon which it's running. The idea is that you schedule queries and record any changes to the host state. Typically, it uses operating system eventing APIs to record changes to files and directories, hardware, networking and others. Note the daemon runs separately from the shell.

Before starting with the daemon, you need to configure the *Osquery* deployment. There are four steps, but we've done the first two. That means we have to:

1. Configure and start the osqueryd service.
2. Manage and collect the query results.

A config file is defined by default in **/etc/osquery/osquery.conf**. This is a JSON file. If you want multiple files, create the folder **/etc/osquery/osquery.conf.d/** and put config files in there.

Configure the config file

A config file is a JSON file ending in **.conf** and looking a bit like the following, although it's been slimmed down a little from the default:

Osquery offers a number of alternative install options.

```
{
  "options": {
    "disable_logging": "false",
  },
  "schedule": {
    "system_info" : {
      "query" : "SELECT hostname, cpu_brand, physical_
memory FROM system_info; ",
      "interval" : 3600
    }
  },
"decorators": {
  "load" : [
    "select ...",
    "select ..."
  ]
}.
"packs" : {
}.
```

The **select ...** are placeholders for full queries. The one under **schedule/system_info** is the main query that gets run at the interval in seconds – 3600 means once an hour. The **decorators** are normal queries that append data to every query. As for **packs**, we'll look at those a bit further on.

The default **osquery.conf** also has another section – **feature_vectors** – but that's only for Windows, so we can ignore it. Much of the information here has been extracted from **https://bit.ly/lxf321wiki**. The values

QUICK TIP

Get familiar with tables by playing with the Osqueryi shell before you start with the Osqueryd daemon. With 154 tables to play with, you'll never be bored.

» INSTALLING OSQUERY

The system we used was Ubuntu 24.04 LTS. On **https://osquery.io/downloads**. You'll see downloads for Windows, Mac and various flavours of Linux. The Debian one downloads a DEB file with EXEs. If you want the installer, scroll further down the page, until you get to Alternative Install Options. However, the first two lines of that script worked OK but the third didn't. A bit of digging found **https://bit.ly/lxf321git** (GitHub issues for *Osquery),* so if that's you, use these instead:

```
$ curl -fsSL https://pkg.osquery.io/deb/pubkey.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/osquery.gpg
$ echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/osquery.gpg] https://pkg.osquery.io/deb deb main" | sudo tee /etc/apt/sources.list.d/osquery.list > /dev/null
$ sudo apt update
$ sudo apt install osquery
```

If your Ubuntu installation is an earlier version, read the GitHub page for some more hints. Once it's installed, you can either use *Osqueryi* for a standalone query or start the Osqueryd daemon.

```
$ sudo cp /opt/osquery/share/osquery/osquery.example.conf /etc/osquery/osquery.conf
$ sudo systemctl start osqueryd
```

To check that it installed OK, run *Osqueryi* from a terminal:

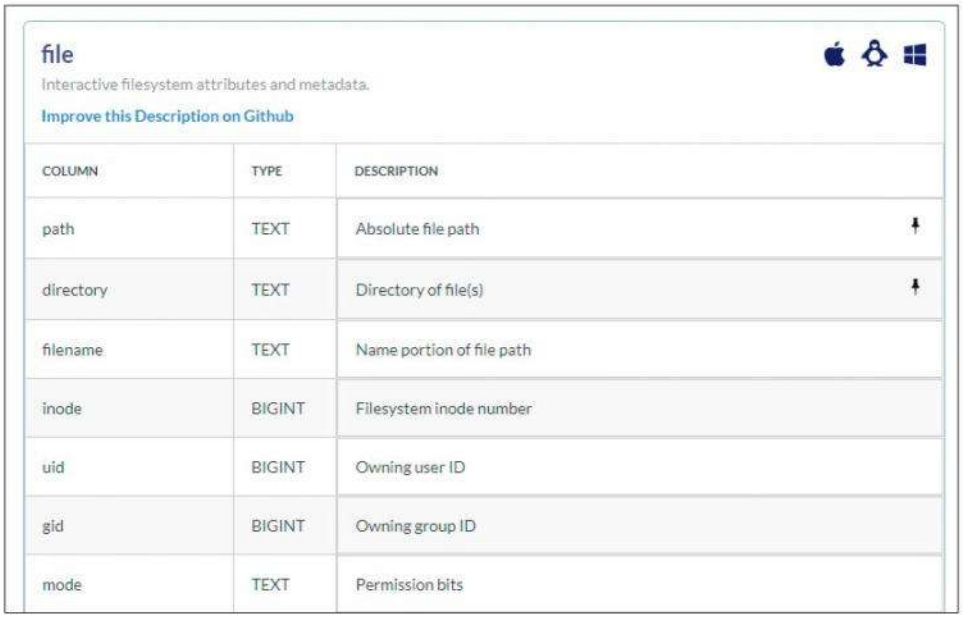
```
>osqueryi
```

If you need help, type `.help`:

```
osquery> .help
```

After that, you'll get a page of commands all starting with a full stop (period).

Try `.features` or `.show`. When you want to exit, either `.quit` or `.exit` will take you back to the terminal.



file

Interactive filesystem attributes and metadata.

Improve this Description on Github

COLUMN	TYPE	DESCRIPTION
path	TEXT	Absolute file path
directory	TEXT	Directory of file(s)
filename	TEXT	Name portion of file path
inode	BIGINT	Filesystem inode number
uid	BIGINT	Owning user ID
gid	BIGINT	Owning group ID
mode	TEXT	Permission bits

A file table definition showing two pushpin fields.

in the **option** section come from the CLI Flags page in the wiki.

The Osquery flagfile

The Google *Flags* application is used by *Osquery* and a flagfile is the recommended way to provide additional flags to osqueryd when it starts up.

```
--flagfile /etc/osquery/osquery.flags
```

The startup options should be put one per line. For instance, if you wanted performance limiting, you would add a line:

```
--watchdog_level=0
```

Use **1** for restricted and **-1** for disabled. There are a lot more flags (over 230 in the **osquery_flags** table) for the osqueryd (as well as 12 for the shell). You can find details about the flagfile on the webpage at **https://bit.ly/lxf321flag**.

QUICK TIP

If you've not used *SQLite* before, practise with an online IDE for running SQL queries. There are several available and they're usually free, such as https://sqliteonline.com or https://sqlime.org.

More on the schedule

The schedule is the heart of osqueryd. In the example above, the name **system_info** is the name given to that scheduled query but you can have multiple queries, each with their own name.

There are a few other parameters that you can set, including restricting it to versions of *Osquery* or have it take a snapshot, which affects how much of the results are stored. If **snapshot** is **false**, then it only stores things that have changed (differentials). If the query might stretch your system resources, set the **denylist** option to **true,** so *Osquery's* watchdog will denylist it – in other words, it will no longer be run.

What are packs?

A pack is a set of queries that are related. By default, there are 10 packs installed when *Osquery* is installed. These are installed into **/opt/osquery/share/osquery/packs** and each is a config file. These are listed but commented out in the default config file, which is where you can enable them. Three of them are useless anyway, as they are for Windows and Mac OS X. List of packs:

- osquery-monitoring
- incident-response
- it-compliance
- osx-attacks
- vuln-management
- hardware-monitoring

Looking at **incident-response**, it has 35 queries that are run either hourly (3600) or daily (86400). However, many of these are for the Darwin-only platform (Mac OS X), Posix or Linux. So, if this pack is enabled and run on a Posix-compatible platform, then the queries with Posix platform will run. Each query in a pack has both a description that explains what it does and a value, explaining what the point of it is.

Here's the query named **kernel_modules** that runs once an hour:

```
  "kernel_modules": {
    "query" : "select * from kernel_modules;",
    "interval" : "3600",
    "platform" : "linux",
    "version" : "1.4.5",
    "description" : "Retrieves all the information for the current kernel modules in the target Linux system.",
    "value" : "Identify malware that has a kernel module component."
  },
```

Discovery queries

This is a way to conditionally decide which packs are run on a host and this depends on a condition being true. For instance, you might only want a pack's queries to be run if the host is running *Docker*.

To do this, you specify one or more SQL queries in a top-level discovery keyword in a pack:

```
{
  "discovery": [
```

```
osquery> select count (*) from file where path like '/home/david/%';
+-----------+
| count (*) |
+-----------+
| 20        |
+-----------+
osquery> select count (*) from file where directory like '/home/david/%';
+-----------+
| count (*) |
+-----------+
```

File and folder counts in Osquery.

» WHAT IS AN SQL QUERY?

If you are familiar with relational databases such as *MySQL, MariaDB, PostgreSQL, SQLite* or even *SQL Server,* or know SQL, you can skip this section.

Relational databases are built from a collection of tables. A table is just like a spreadsheet. It's a collection of data defined by columns and rows, where every value in a column is of the same type. One column might hold an integer ID, another a name string and another a date. You access the data in rows. If this table were a payroll table, it might have a row for each employee, like this:

Id	Name	StartDate
1	David Bolton	04/07/2017
2	Fred Smith	03/01/2012
3	Daphne Ward	17/11/2023

To get values out of such a table, we do a select query like this one, which returns all employees who started work on or after 1st January 2016:

```
Select * from payroll where startdate >= '01/01/2016';
```

That would return the first and last row but not the second one, because the 2012 date is before 01/01/2016. The * in the select returns all rows. If you just want the name and start date, then you would use:

```
Select Name,startdate from payroll where startdate >=
'01/01/2016';
```

This would return the second and third columns from the first and last rows. Note: you must always put a semicolon on the end of SQL queries. The version of SQL used is the *SQLite* version; you can find the definition at **www.sqlite.org/lang.html**.

```
    "select pid from processes where name ='Docker;'
  ]
   "queries": {
  ...
   }
}
```

If this returns 0 rows, then the pack's queries are not run. Note: we've used a simple **Name =** comparison but you might find doing a **like** with % wildcards is better. Try the query using *Osqueryi* to figure out the query to find *Docker* or whatever process you are looking for.

Logging options

There are several plugins provided by *Osquery* to specify where the logs go. By default, it uses the filesystem and logs are written as JSON to specific file paths. Other logger options include tls, syslog and some Amazon AWS.

The page at **https://bit.ly/lxf321log** documents the various command-line flags that affect logging. There are simply too many to list here. There are two types of logs: status logs and query schedule results logs.

The status logs are generated by the Glog logging framework. By default, the osqueryd daemon sends **INFO**, **WARNING**, and **ERROR** logs to the configured logger plugins and to the process's stderr. You may configure this behaviour using several flags documented in **https://bit.ly/lxf321cli**. Results logs look like the following output:


```
{"name":"system_info","hostIdentifier":"david-
PC8","calendarTime":"Sun Aug 25 09:51:14 2024 UTC",
"unixTime":1724579474,"epoch":0,"counter":0,"nume
rics":false,"decorations":{"host_uuid":"d48b276d-
530a-4bca-8242-3a8fdeab644e","username":""},"colu
mns":{"cpu_brand":"11th Gen Intel(R) Core(TM)
i7-11700K @ 3.60GHz","hostname":"david-
PC8","physical_memory":"13026881536"},"action":"ad
ded"}
```


Going further

This is a very impressive piece of open-source software. More so because it's cross-platform and very well documented.

It's easy to get it installed but we feel we've barely scratched the surface of it. For instance, we didn't get to try log aggregation – dealing with the logs once they're produced or even analysing them. For more advanced use, you can learn how to create virtual tables, do remote logging to Amazon AWS Kinesis Streams and Kinesis Firehose.

And for even more advanced use, you can learn about the *Osquery* SDK (**https://bit.ly/lxf321sdk**) and write extensions for it. Extensions are commonly written in C++, but can also be developed in Python, Go or any language really that supports the Apache Thrift framework (**https://thrift.apache.org**). Extensions are separate processes that communicate over a Thrift IPC channel to the *Osquery* core in order to register plugins or virtual tables. LXF

QUICK TIP

To get a feel for queries, look at the pack files, load each one up into a text editor and examine the source file. It's not just the SQL but the description, platform and value that turn a simple query into a great one.

```
osquery> select * from osquery_flags where shell_only=1;
+-------------+--------+-----------------------------------------------------+---------------+-------+------------+
| name        | type   | description                                         | default_value | value | shell_only |
+-------------+--------+-----------------------------------------------------+---------------+-------+------------+
| connect     | string | Connect to an extension socket                      |               |       | 1          |
| csv         | bool   | Set output mode to 'csv'                            | false         | false | 1          |
| extension   | string | Path to a single extension to autoload              |               |       | 1          |
| header      | bool   | Toggle column headers true/false                    | true          | true  | 1          |
| json        | bool   | Set output mode to 'json'                           | false         | false | 1          |
| json_pretty | bool   | Set output mode to 'json_pretty'                    | false         | false | 1          |
| line        | bool   | Set output mode to 'line'                           | false         | false | 1          |
| list        | bool   | Set output mode to 'list'                           | false         | false | 1          |
| pack        | string | Run all queries in a pack                           |               |       | 1          |
| planner     | bool   | Enable osquery runtime planner output               | false         | false | 1          |
| profile     | int32  | Enable profile mode when non-0, set number of iterations | 0        | 0     | 1          |
| separator   | string | Set output field separator, default '\|'            | \|            | \|    | 1          |
+-------------+--------+-----------------------------------------------------+---------------+-------+------------+
```

The **small set of osqueryi shell flags.**

Part Six! Don't miss next issue, subscribe on page 16!

LXF SHELL

Credit: https://github.com/fritzone/lxf-shell

Add some history to the LXF Shell

Refusing to ever learn from anything, **Ferenc Deák** realises that adding history to the LXF Shell will help him repeat his mistakes.

OUR EXPERT

Ferenc Deák feels this project is his Sistine Chapel, only with less of the Pope shouting at him and more Mancunian editors.

QUICK TIP

The code for the shell still can be found at https://github.com/fritzone/lxf-shell.

In the world of the Linux shell, the command line history stands as a silent but powerful ally to users navigating the intricate pathways of their systems. It serves as a virtual breadcrumb trail, providing users with a means to recall, analyse and replicate their past interactions with the system.

In this episode of our shell saga, we will delve into the mechanisms, practical applications and, last but not least, the implementation details of how to make the command line history work as expected by a grown-up shell. Not that ours is one, but it will be.

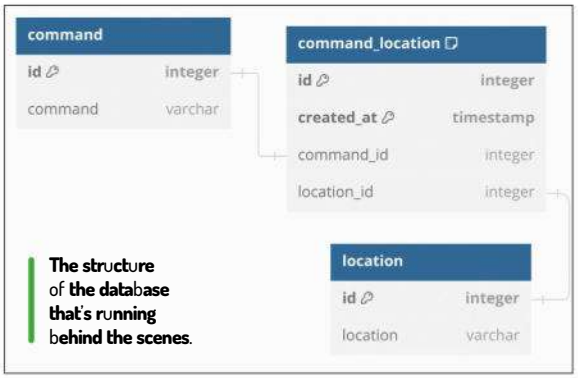
The mechanisms

Modern shells, such as *Bash* and *Zsh,* offer similar command history navigation using the up and down arrow keys, Ctrl+R for reverse search, and more advanced commands such as `!n` , `!!` and `!string` to execute specific or recent commands. For the implementation of this feature in our tiny shell, we plan to provide support for the previous and next elements. However, as it is not that difficult, we will also add some extra features: if the user presses Ctrl+Up/Down, we will search for the history in the current folder.

For the moment, we present the following structure for these hotkeys, without too much explanation – but no worries, that will come at a later stage:

```
static const std::map<const char*, std::vector<uint8_t>>
reservedKeys {
   {"Up", {27, 91, 65}},
   {"Down", {27, 91, 66}},
   {"CtrlUp", {27, 91, 49, 59, 53, 65}},
   {"CtrlDown", {27, 91, 49, 59, 53, 66}},
};
```

Current shells usually keep the history in a file called **.xxx_history**, where **xxx** is based on the name of the shell, for example **.bash_history**. These are simple text files that contain all the commands executed since the shall was taken into first use. As our solution will be a bit more complex, we need a quick and dirty database set up for this, so a short intro to *SQLite* is to be found in the boxout *(opposite).* We will create the following database layout to store the commands. The following SQL script is used to create the database:



The structure of the database that's running behind the scenes.

```
CREATE TABLE command ( id INTEGER PRIMARY
KEY, command VARCHAR(255) );
CREATE TABLE location ( id INTEGER PRIMARY KEY,
location VARCHAR(255) );
CREATE TABLE command_location (
 created_at DATETIME,
 command_id INTEGER,
 location_id INTEGER,
 PRIMARY KEY (created_at),
 FOREIGN KEY (command_id) REFERENCES
command (id),
 FOREIGN KEY (location_id) REFERENCES location
(id)
);
```

It might feel overcomplicated to have three tables, but we have to keep in mind, that:

- A command can appear more than once;
- A directory (location) can appear more than once;
- And to keep the database in the most optimal shape (aka the third normal form), we should avoid all data duplication and keep everything based on unique keys.

So, the above restrictions automatically impose the required structure for the most optimal database.

Overriding getline

Up to this point, our friend `std::getline` was responsible for handling the input of the command to be executed by the shell from the standard input. Unfortunately, `std::getline` can't be used to notify us whether any of

the specific key combinations were pressed or not, because it is not in the nature of the command to do it.

Correctly identifying the required key combinations is an entirely different task and it requires a different approach. We need to dig deep, to the level where the terminal handles the raw key inputs, and fetch the data directly from there, so we advise you to consult the boxout about low-level terminal handling *(page 97)*.

Raw or cooked

We plan to hijack the input and read the keypresses in directly, in order to properly act upon them, but for that we need to set the terminal in raw mode, through the **termios** structure. This provides detailed control over terminal I/O characteristics. Here's a simplified approach to switch between cooked and raw modes:

```
void enableRawMode() {
   struct termios raw;
   tcgetattr(STDIN_FILENO, &raw);
   raw.c_lflag &= ~(ECHO | ICANON);
   tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);}
void disableRawMode() {
   struct termios term;
   tcgetattr(STDIN_FILENO, &term);
   term.c_lflag |= (ECHO | ICANON);
   tcsetattr(STDIN_FILENO, TCSAFLUSH, &term);}
```

The above two functions turn the raw mode of the terminal on and off, enabling us to read in the key data as required, byte by byte, and then perform the required action based on what key was pressed. We have also included a small function (**keyscanner.cpp**, to be found on the project's GitHub site) to read in the raw bytes of the input and properly write out the correct sequence to handle the specific key combinations.

For the standard keys, there is a more-or-less standard escape sequence; however, more exotic key combinations, such as pressing keys with Ctrl or Alt modifiers, might result in different escape sequences than the one on our computer. So, if you experience any inconsistencies with these, please get in touch in order to include your key combinations in the escape sequence container, too.

Low-level reading

In order to properly handle the reading of keystrokes from stdin, we will employ the **select** and **read** functions available in Linux. As an introduction, the following piece of code will be used to read in the user's input from the keyboard, byte by byte:

```
std::vector<uint8_t> read_stdin() {
   std::vector<uint8_t> result;
   uint8_t buffer[256] = {0};
   int size = read(STDIN_FILENO, buffer, sizeof(buffer) -
1);
   if (size == -1) return {};
   else {
      for (int i = 0; i < size; ++i) result.push_
back(buffer[i]);
      return result;
   }
}
```

The function can't get simpler than this, it just uses the **read** system call to read in data from **STDIN_FILENO** and build up a vector from it, then return the vector. The underlying Linux system is responsible for filling up the stdin with the required bytes according to the keypress of the user. Before calling this function, some preparation is needed. The **select** system call in Linux is a powerful tool used to simultaneously monitor multiple file descriptors, checking if they are ready for some kind of I/O operation. A file descriptor is a plain integer, with the role of a handle used in Linux to represent an open file or a resource. **Select** is used in network programming and other scenarios where you need to handle multiple input/output streams at once, but for us it is perfect for handling input reading, too.

The **fd_set** data structure is used to represent a set of file descriptors. Before calling **select**, you typically initialize **fd_set** to specify which file descriptors you're interested in. Some of the code demonstrating the use of these calls in the **LXF** Shell is presented below:

```
struct termios orig_termios;
tcgetattr(STDIN_FILENO, &orig_termios);
enableRawMode(orig_termios);
fd_set readfds_orig = {0};
FD_SET(0, &readfds_orig);
int max_fd = 0;
while(true) {
   if(currentPrompt.empty()) std::cout << "\rlxfsh$ " <<
command;
   else std::cout << "\r" << currentPrompt  << "\e[0m" <<
command;
   fflush(stdout);
   fd_set readfds = readfds_orig;
   if (select(max_fd + 1, &readfds, 0, 0, 0) != -1 && FD_
ISSET(0, &readfds)) {
```

» SQLITE CRASH COURSE

Databases are essential tools for storing, managing and retrieving data efficiently – `sudo apt install libsqlite3-dev` is a notable example of a database management system that is widely appreciated for its simplicity, reliability and ease of integration, and which reads and writes directly to ordinary disk files. To start with *SQLite*, you have to install it if your system does not has it by default: `sudo apt install sqlite` . Then you can create a new database (or open an existing one) by specifying the filename: `sqlite3 mydatabase.db` . From this point on it is just crafting simple or more complex SQL commands, with the observation that *SQLite* supports only a subset of the full SQL syntax – there are a lot of online resources to help.

The good news for grumpy C++ developers is that it is very easy to use *SQLite* from a C (or C++) codebase, because there is already a library written for this purpose: `sudo apt install libsqlite3-dev` (or your distro -specific one) will install the *SQLite* library. *SQLite* also has support for *CMake*:

```
find_package (SQLite3)
if (SQLITE3_FOUND)
  include_directories(${SQLITE3_INCLUDE_DIRS})
  target_link_libraries (${PROJECT_NAME} ${SQLITE3_LIBRARIES})
endif (SQLITE3_FOUND)
```

Armed with this knowledge, we can now move on further down the line and invoke the *CMake* support in the form of `\#include <sqlite3.h>` and then use `sqlite3_open(dbPath, &db);` to load a database. From this point on, `sqlite3_exec` is used to execute SQL queries on the database, and when done with all the work, `sqlite3_close` should be used to close the connection. For readers interested in the subject, the page **https://sqlite.org/cintro.html** contains a quick introduction to how to use *SQLite* from C or C++.

```
    auto keys = read_stdin();
    if(keys.size() == 1 && keys[0] == 10) break;
    if(keys.size() == 1 && (isprint(keys[0]) ||
isspace(keys[0]))) command += keys[0];
    else {
      std::string shortcut = "";
      for (const auto& kv : reservedKeys) {
        if (kv.second == keys) {
          shortcut = kv.first; break; } }
      if(!shortcut.empty()) {
        command = resolveHistory(shortcut);
      } } } }
disableRawMode(orig_termios);
```

This simple code tries to imitate primitive input handling, but for now it does just three key things:

1. If the user presses Enter (`keys.size() == 1 && keys[0] == 10`), it considers the current command completed and passes the value out to the shell, handling it.
2. If the current key can be added to the current command, such as a character, number or symbol (`keys.size() == 1 && (isprint(keys[0]) || isspace(keys[0]))`) it appends it to the current command.
3. The interesting part is the sequence where the code tries to identify whether the keypress is one of the reserved sequences, and if yes, it replaces the command with the relevant entry from the history.

Now, the **reservedKeys** structure from the start gains some meaning, and we can identify the various byte sequences (aka escape sequences) from it.

For the purposes of this article, we will leave this crippled input handling, because properly handling all the required keys, such as Home, End, Delete and Backspace, and adding their proper implementation, having in place the insertion, overwrite features and so on, could easily turn into an endless mess for which there's no place in an educational article. The only thing remaining unimplemented for us is the history feature.

History and historians

There are two important features regarding the history:

1. It makes possible adding and retrieving commands relatively easily using basic *SQLite3* functionality.
2. It repeats itself.

There is nothing we can do concerning the second, but handling an *SQLite* database using the C API provided by the developers consists of long and stuffy code sequences, so we will present only the larger picture of what happens behind the scenes. (If you're interested, however, check out the tutorial at **www.sqlite.org/cintro.html**.)

The first operation we need to do is insert elements in the database – this is done using the following straightforward algorithm:

1. When the user presses the Enter key, we call the **storeCommandHistory** function.
2. The function checks whether is there an existing entry for the command in the command table. If yes, it uses the ID from it, otherwise it inserts a new entry in the database and uses the returned ID.
3. Sees if there is an entry for the given location in the location table. If yes, it use its ID, otherwise it creates a new entry for it, and uses the returned ID (for our specific purpose, the location is provided by `std::filesystem::current_path()`).
4. With the two ID values from above, prepare a query to insert the actual history element in the **command_location** table.
5. And from this point on move to the execution of the command, as in previous steps.

The command retrieval is a much more intricate process, however. To properly retrieve the values, we need to keep a counter, which is used to retrieve various command and location combinations from the **command_location** table. If the counter is 0, it returns the last element from the **command_location** table, if it is 1, it returns the penultimate element, and so on...

For the simple case – the generic history, not bound to the confines of a directory – the following function will do the trick:

```
std::tuple<std::string, std::string>
getCommandLocation(sqlite3* db, int counter) {
  std::tuple<std::string, std::string> result;
  std::string query =
    "SELECT command.command, location.location "
    "FROM command_location "
    "JOIN command ON command_location.
command_id = command.id "
    "JOIN location ON command_location.location_id =
location.id "
    "ORDER BY command_location.created_at DESC "
    "LIMIT 1 OFFSET " + std::to_string(counter) + ";";
  executeQuery(db, query, commandLocationCallback,
&result);
  return result; }
```

The function is not overly complicated – it creates an SQL string built up to suit our needs:

- The `ORDER BY command_location.created_at DESC` sorts the results in descending order by the timestamp, ensuring that 0 returns the most recent entry, 1 the second most recent, and so on.
- The `LIMIT 1 OFFSET` counter retrieves a single row based on the given offset (the counter).

And then it calls the **executeQuery** with a specific callback method to build up the resulting tuple, and the address of the tuple:

```
int executeQuery(sqlite3* db, const std::string& sql, int
(*callback)(void*, int, char**, char**), void* data) {
  char* errorMessage = nullptr;
  int rc = sqlite3_exec(db, sql.c_str(), callback, data,
&errorMessage);
  if (rc != SQLITE_OK) {
    std::cerr << "SQL Error: " << errorMessage <<
std::endl;
    sqlite3_free(errorMessage);
    throw std::runtime_error("Failed to execute SQL
query"); }
  return rc; }
```

This is a bit more interesting, since it goes a bit deeper into the inner workings of the *SQLite* API, and we can see how the data is passed through:

- The **commandLocationCallback** function (presented below) serves as the callback for **sqlite3_exec**, extracting the command and location values, and storing them in a tuple (which is to be found at the location data in the memory).
- The **executeQuery** function runs the query and uses a callback to process the result.

The **commandLocationCallback** function looks like:

```
int commandLocationCallback(void* resultTuple, int
argc, char** argv, char** colNames) {
```

```
  if (argc != 2) throw std::runtime_error("Unexpected number of columns");
  std::tuple<std::string, std::string>* result = static_cast<std::tuple<std::string, std::string>*>(resultTuple);
  std::get<0>(*result) = argv[0] ? argv[0] : ""; // command
  std::get<1>(*result) = argv[1] ? argv[1] : ""; // location
  return 0; }
```

The function starts by checking if **argc** (the number of columns returned) is exactly 2. This is crucial because the function expects exactly two columns: one for the command and one for the location. If the number of columns is not what's expected, it throws a runtime error, otherwise it continues to the next stage.

The function casts the **void*** pointer (**resultTuple**) to a pointer to a **std::tuple<std::string, std::string>**. It then retrieves the command and location values from the **argv** array, checking if each is **nullptr** (which would indicate a NULL value in the database). If **nullptr**, it assigns an empty string; otherwise, it assigns the string value from the database.

These values are stored in the tuple at the specified indexes: index 0 for command and index 1 for location. The function returns 0 to indicate successful execution. In *SQLite's* callback system, returning a non-zero value typically indicates an error, causing **sqlite3_exec()** to terminate.

All this falls into place with the **resolveHistory** function, specifically the implementation of the previous command, triggered by the Up key:

```
static int currentHistoryElementIndex = 0;
std::string resolveHistory(const std::string& shortcut) {
  if(shortcut == "Up") {
    sqlite3* db = nullptr;
    int rc = sqlite3_open("lxf-shell-history.db", &db); // Open or create the database
    if (rc) {
      std::cerr << "Can't open database: " << sqlite3_errmsg(db) << std::endl;
      return ""; }
    auto prevCommand = getCommandLocation(db, currentHistoryElementIndex);
    sqlite3_close(db);
    currentHistoryElementIndex ++;
    return std::get<0>(prevCommand); }
  return ""; }
```

Again, this is not black magic– the function just opens the database and the previous command is fetched, using the method above (or nothing if the **currentHistoryElementIndex** is out of the bounds). And that's it. The function returns the previous command to the input handler, and that updates what needs to be updated.

We also mentioned a specific extension of the shell for our small delight: if we press Ctrl+Up we retrieve the previous command from the history that was executed in our current working directory. The implementation of this feature is very similar to **getCommandLocation** except that it also needs to take into consideration the current location we are in (again provided by `std::filesystem::current_path()`). The main difference is the SQL query, which also takes into consideration the location where we are in the directory tree. The code, again, is too long to fit in here, but the project can be found at **https://github.com/fritzone/lxf-shell** . Please go,and read up on the code – it contains a lot of interesting details.

The future is tabbed

Before we conclude this episode, a few closing words: the code right now is unstable, very sensitive and tends to throw toddlerish fits whenever it feels it helps to achieve its mysterious goals. The purpose of the example programs at this stage is not to provide a rock solid shell with which you can replace your current one, but more of a sandbox, where we can experiment and learn how to do unusual stuff and push the limits of Linux. Stay tuned for the next episode, where we will show how to implement command line completion. LXF

» LOW-LEVEL TERMINAL HANDLING

Terminals usually operate in one of two modes: cooked or raw. Cooked mode is the default. In this mode, the OS provides a level of processing to handle special characters (such as Backspace, Ctrl+C and Enter) and buffers input until the Enter key is pressed. This mode simplifies basic input tasks, but due to its limitation, it is very hard to get feedback when the arrow keys are pressed, for example.

In raw mode, the terminal handling is passed almost directly to the application, with minimal intervention from the OS. This mode disables line buffering and special character processing, giving the application full control over every byte entered by the user. Raw mode is crucial for apps that require real-time keystroke handling, such as text editors, shell environments and – why not? – interactive games, because everyone loves to play *Doom* on their Linux console, courtesy of **https://github.com/wojciech-graj/doom-ascii**.

Escape sequences are character sequences that start with an escape character (often represented as `ESC` or `\x1b`) followed by a series of characters that instruct the terminal to perform specific actions. These actions include moving the cursor, changing text colour, clearing portions of the screen, and much more. For example, when the arrow keys pressed, the terminal emits an escape sequence rather than a simple character. So, pressing the up arrow might generate the sequence `\x1b[A` or more numerically `27, 91, 65`.

NEXT MONTH

LXF322 will be on sale Tuesday 12th November 2024

POP ON YOUR FEDORA 41

Well into middle age, we get you up and running with the workstation OS your PC deserves!

Browser Wars 2025

Which is the best web browser to take into next year? We pit the biggest names against each other.

Expand your memory

Get to grips with Linux memory handling and grab yourself some faster and bigger RAM to boot.

Infrared photography

Dust off that old kit as we revitalise the old technique of snapping moody infrared landscape images.

Linux drivers

Master Linux hardware drivers and maybe even get that old printer working – it's not as hard as you think!

Content of future issues subject to change. Did the car kill off hat wearing?

CREDIT: Magictorch

LINUX FORMAT
The #1 open source mag

Future Publishing Limited, Quay House, The Ambury, Bath, BA11UA
Email contact@linuxformat.com

EDITORIAL
Editor-in-chief Neil Mohr
Art editor Fraser McDermott
Production editor Katharine Davies
Content director Marc Chacksfield
Group art director Warren Brown
Editorial contributors
Mike Bedford, Jonni Bidwell, David Bolton, Neil Bothwick, Kerry Brunskill, Stuart Burns, Ferenc Deák, Nate Drake, Tam Hanna, Brandon Hill, Dave James, John Loeffler, Jon Masters, Nick Peers, Les Pounder, Michael Reed, Mayank Sharma, Shashank Sharma, Michelle Rae Uy
Cover illustration Magictorch.com

Content production Adequate Media Limited

ADVERTISING
Commercial sales director Clare Dove
clare.dove@futurenet.com
Advertising director Lara Jaggon
lara.jaggon@futurenet.com
Account director Andrew Tilbury
andrew.tilbury@futurenet.com

INTERNATIONAL LICENSING
Head of print licensing Rachel Shaw
Linux Format is available for licensing and syndication.
To find our more contact us at licensing@futurenet.com
or view our content at www.futurecontenthub.com

NEW SUBSCRIPTIONS & PAST ISSUES
Web www.magazinesdirect.com

EXISTING SUBSCRIPTIONS
Web www.mymagazine.co.uk
Subscription delays: Please allow up to seven days before contacting us about a late delivery to **help@magazinesdirect.com**

MANAGE YOUR SUBSCRIPTION ONLINE WITH MYMAGAZINE
Visit **www.mymagazine.co.uk/FAQ** to view frequently asked questions or log in at **www.mymagazine.co.uk**

CIRCULATION
Newstrade & retail category director Ben Oakden

PRODUCTION AND DISTRIBUTION
Group head of production Mark Constance
Production manager Nola Cokely
Senior ad production manager Jo Crosby
Digital editions manager Jason Hudson

THE MANAGEMENT
Managing director technology group Paul Newman
Global head of design Rodney Dive
Commercial finance director Tania Brunning
Printed by William Gibbons & Sons
Distributed by Marketforce UK
121-141 Westbourne Terrace, London W2 6JR. **www.marketforce.co.uk**
For enquiries email: **mfcommunications@futurenet.com**

Order and access past issues: If you are an active subscriber, you have instant access to past issues through your iOS or Android device/s. Your digital magazine entitlement is available at no additional cost and no further action is required. Pocketmags library may not have access to the full archive of digital past issues. You will only be able to access the digital back issues as long as you are an active subscriber.

To purchase single past issues (print format only): Visit www.magazinesdirect.com (click on Single Issues tab) or email help@magazinesdirect.com. Magazinesdirect.com is owned and operated by Future Publishing Limited.

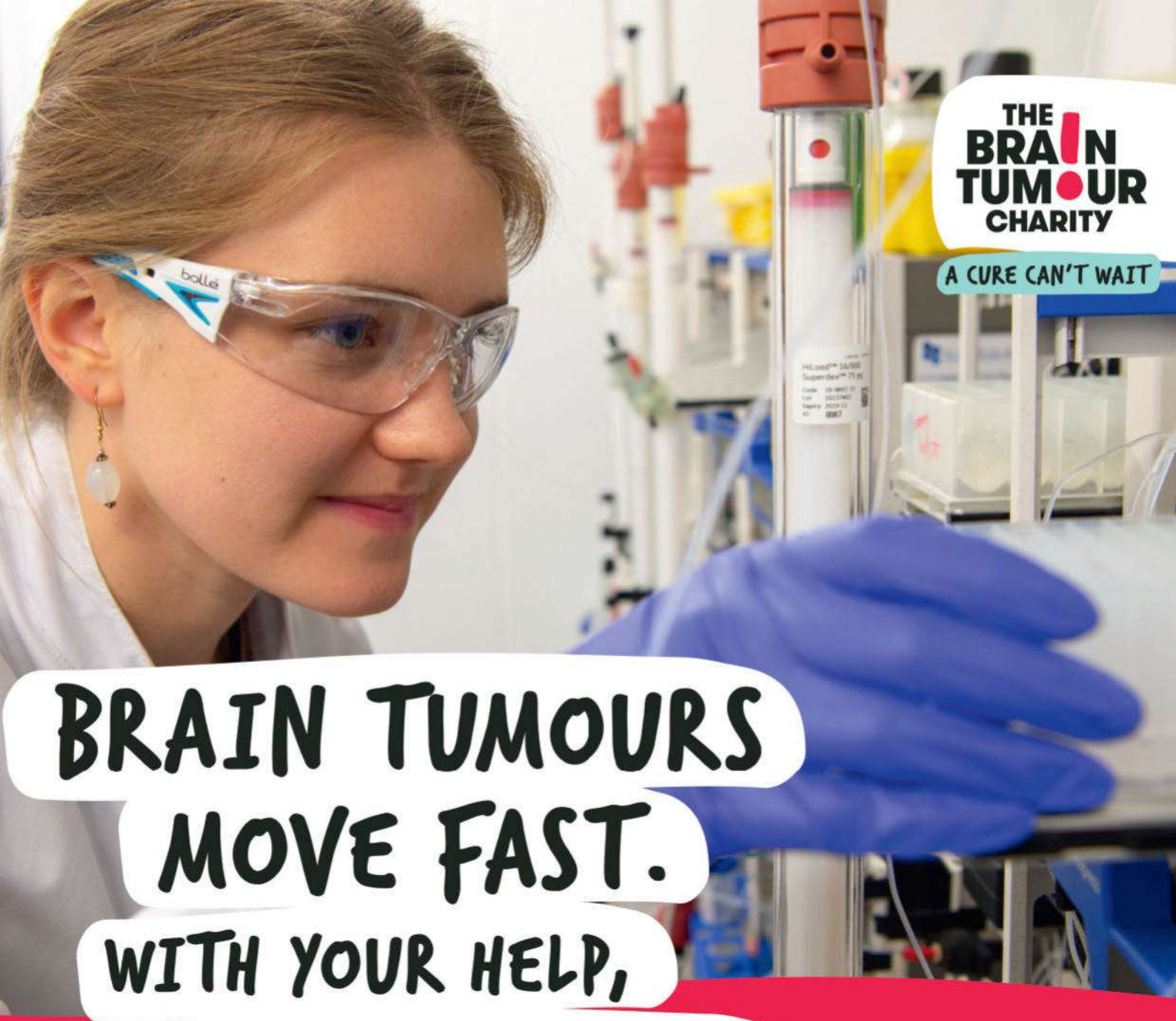
BAR rates for Linux Format: £84.37 for UK, €171 for Europe, $194 for USA, £149 for rest of world.

The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

EFF.ORG

ELECTRONIC FRONTIER FOUNDATION

Protecting Rights and Promoting Freedom on the Electronic Frontier



THE BRAIN TUMOUR CHARITY

A CURE CAN'T WAIT

BRAIN TUMOURS MOVE FAST.

WITH YOUR HELP, WE CAN TOO!

We're working to create a future where brain tumours are curable.
We urgently need your help to accelerate research.

Text DEFEAT5 to 70507 to donate £5, please help us to find a cure.

thebraintumourcharity.org

Registered with FUNDRAISING REGULATOR